12-2014

Implementing Lensless Cameras in Autonomous Robotic Systems

Aleksandar Tomović
St. Cloud State University

Implementing Lensless Cameras in Autonomous Robotic Systems

by

Aleksandar Tomović

A Starred Paper

Submitted to the Graduate Faculty of

St. Cloud State University

in Partial Fulfillment of the Requirements

for the Degree

Master of Science

December, 2014

Starred Paper Committee:

Andrew Anda, Chairperson
Jayantha Herath
Yi Zheng

ABSTRACT

The open-source Robot Operating System (ROS) is a mixed and scalable P2P network-based robotics framework. We examine lensless compressive imaging using a hardware apparatus assembly having an imaging sensor, but no lens. Cameras with lenses have been the standard, but several factors constrain their application. Lensless cameras may reduce the cost, size, and weight of image processing as we move away from use of expensive lenses in robot designs. Lensless cameras can be used also in applications such medicine, where apparatus size is very important.

To support our objective we show how ROS applications are developed and most importantly how one can build applications that allows users to complete useful tasks in a timely manner with high performance.

ACKNOWLEDGMENTS

I would like to thank my commitee members Professors Jayantha Herath, Yi Zhang and Andrew Anda for their continuous support. I have to thank my project sponsor Stone3000 Inc. Stone3000 inc. financed this project, provided material; and provided the lab environment needed for my experiments. Special thanks to St. Cloud State University, Computer Science Department and faculty, Electrical Engineering Department and faculty, Ms. Mary Thrall, fellow classmates and friends for providing a great learning atmosphere and for providing skills that made a big contribution during this project.

Thanks to my family Andjela, Stefan and Annamaria, for being there when I needed them.

As per NDA agreements with Stone3000 and optical sensor manufacturer Aptina Inc. the codes and data sheets used in these experiments are considered proprietary and are not discussed in this paper.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Chapter I

ROBOT OPERATING SYSTEM

Modern robotics concerns how a robot can perceive the world, make sense of its surroundings, and interact with its environment. We introduce the basics of robotics and autonomy algorithm theory. Open source libraries and inexpensive robot platforms facilitate creating advanced robot capabilities. We present a hands-on introduction to applied robotics software programming. We introduce the Robot Operating System (ROS) robotics framework, including affiliated open source autonomy libraries, all integrated into a small robot equipped with a depth sensor and with camera with lenses and with lensless camera.

INTRODUCTION

The ROS robotics *framework*, also termed a work platform, including affiliated open source autonomy libraries, all integrated into a ground robot equipped with sensors, is sufficient to present robot behaviors, robot manipulation, and robot (P2P) communication as part of our research experiment conducted in the lab at Stone3000 Inc. Applying basic robotic knowledge, robots may now be quickly utilized in an industrial environment to help improve production flow and cost efficiencies. Robotic technology connects information to the physical world around us. Such applications include unmanned ground vehicles, unmanned aerial vehicles, and robots exploiting visual optics that can use the internet to navigate with partial classification. A robot's structure, chassis, geometry, and joints, are all important aspects of robot design. Integranting of kinematics and dynamics, how a robot moves, is essential for a successful design. During our research we adapted an already developed platform available on the market, to perform production tasks. The ROS is supported by full time developers at the Open Source Robotics Foundation (OSFR) and by independent contributors including graduate students, robot builders, application developers, and hobbyists [1].[1] This paper is structured as follows: First we discuss robot communications, Peer-to-Peer networks and network styles of communication. We explain how to develop intelligent robot considering navigation and paths planning algorithms. We continue to discuss path planning with uncertainty, tools, and nonlinear state estimation. We present the navigation problems of autonomous robots, simultaneous localization and mapping.

ROBOT OPERATING SYSTEM (ROS)

The Robot Operating System ROS [2] is a peer-to-peer *robot middleware* package. We use the ROS because it facilitates *hardware abstraction* and *code reuse.* The ROS peer-to-peer networks are heterogenic and scalable. The ROS, being extensible, does not impose a methodology (does not wrap `main()`), thereby encouraging development of ROS-independent libraries. The ROS design goals are

[1] Contributions are coordinated via `ROS.org`, compiling wiki documentation, and the ROS standards [11].

tools-based using a *microkernel* .The ROS do not use a monolithic design. In computer science, a microkernel (also termed μ-kernel or Samuel kernel) is the near-minimum amount of software that can provide those mechanisms necessary to implement an operating system (OS). These mechanisms include low-level address space management, thread management, and inter-process communication. The ROS is multi-lingual (C++, Python, Java, Lisp, MATLAB, Lua, etc...). Open source, the ROS is released under the terms of the Berkeley Software Distribution License BSD [3] license, and this software is free for non-commercial and research use. The BSD licenses are a family of permissive software licenses, imposing minimal restrictions on the redistribution of covered software. The `ros-pkg` contributed packages are licensed under a variety of open source licenses.

In the ROS, each function is decomposed into a number of nodes that communicate with each other, and the functions typically run as separate processes. Communication between nodes is controlled by the ROS master.

The modularity of the ROS decomposes the complex system into simpler specific independent tasks. The ROS modularity executes in serial or parallel and can execute on one or more machines. Data communication is performed between software modules.

ROBOTIC NAVIGATION AND EXPLORATION UNDER UNCERTAINTY

Building *intelligent* robots that can accomplish tasks without human help has fascinated many, especially those in the artificial intelligence community. From a technical point of view, autonomous robot navigation focuses primarily on the design of robots having abilities such as generating optimal global paths to maneuver themselves to a target location in a real time environment. An autonomous robot can reactively correct its course by circumventing obstacles trough collision avoidance and the exploration of unmapped regions.

Global Path Planning

Global path planning (GPP) addresses autonomous robot navigation in contexts including unmanned ground vehicle Control [4], an unmanned aircraft or Rotorcraft [5], and the Mars Rover [10]. A *robot global path planning* (RGPP) system senses the information from the environment and plans a collision free trajectory to navigate to a destination subject to physical constraints. Initially, GPP algorithms assumed a robot to have complete knowledge of its environment and its own placement. However, in real applications, information is generally only partially available or unavailable in advance. Therefore, more recent work has focused on how to generate a global path in the presence of sensor noise and map incompleteness.

Classical Path Planning

Classical path planning (CPP) usually represents the world using the termed the configuration space [4], [1]. The configuration spaces in classical mechanics are generalized coordinates (parameters that define the configuration of a system) and a vector space is defined by those coordinates. Methods which transform the configuration space into cell regions that can be used for path planning are termed *cell decomposition methods*.

Cell decomposition methods, tiling the configuration space into convex polygons, termed cells, use path search methods to search through cells to find the optimal path to the goal. Cell decomposition methods are the most popular approaches to many of applications in robotics. *Roadmap methods* fill the configuration spaces with roadmaps/graphs that contain nodes representing reachable robot configurations and edges which are one-dimensional curves representing the free space between the nodes corresponding to topographical properties. Roadmap methods for finding shortest paths include visibility graphs, which connect the nodes of polygonal obstacles, and Voronoi roadmaps which use borders as edges [3]. Other methods in use are probabilistic roadmap methods (PRM), potential field methods (PFM), and harmonic potential field map methods (HPFM). Robotic systems can achieve global planning and avoid being trapped in local minima by integrating the methods of certainty grids used for obstacle representation and potential fields devised for navigation and by considering the entire path [7].

Path Planning with Uncertainty

The *classical path planning* is generally performed by robots in fully observable environments with conditions usually assumed to be deterministic and discrete. Probability based frameworks, augumented by statistical tools (extended Kalman filter (EKF) [8] and particle filter–sequential Monte Carlo [9]), have been developed starting in the 1980s. In estimation theory, the EKF is a nonlinear variant of the

Kalman filter, which linearizes about an estimate of the current mean and covariance. In the case of defined transition models, the EKF has been considered the standard in the theory of nonlinear state estimation, navigation systems and GPS. The *particle filters* or *sequential Monte Carlo* (SMC) methods consist of a set of on-line[2] posterior density estimation algorithms that estimate the posterior density of the state-space by directly implementing Bayesian recursion equations. SMC methods use a grid-based approach, using a set of particles to represent the posterior density. These filtering methods make no restrictive assumptions regarding the dynamics of the state-space or the density function. SMC methods provide a well-established methodology for generating samples from the required distribution without requiring assumptions about the state-space model or the state distributions [9].

Simultaneous Localization and Mapping (SLAM)

The navigation problem of autonomous robots is solved by frameworks building a map while a robot continually localizes itself is termed simultaneous localization and mapping (SLAM)[3]. SLAM [3] is a robotics research sub-field which solves the problems of building a map of the environment where in the robot is locating and localizing itself continually. Mathematically, SLAM estimates the full posterior distribution over robot poses and landmark locations in a recursive fashion

---

[2] The on-line algorithm receives one example at the time and maintains a parameter that is essentially an average of the past examples [25].

[3] SLAM is a process by which a vehicle can build a map of an environment and at the same time use the map to know its location. In SLAM, both the trajectory of the platform and the location of all landmarks are estimated online with no priori knowledge of location.

over time, given sensor readings corrupted by noise and systematic errors. The posterior distribution however, carries the benefit of increased robustness. The need to approximate arises from the fact that most robot worlds are continuous. Computing an exact posterior distribution[4] is typically infeasible, since distributions over the continuum possess infinitely many dimensions. Sometimes, one is fortunate in that the uncertainty can approximated tightly with a compact parametric model (e.g., discrete distributions or Gaussians); in other cases, such approximations are too basic and more complex representations must be employed.Therefore, SLAM is a nonlinear filter.

Local Navigation

Local Path planning is sensor-centered reactive, fast response system which assumes incomplete knowledge of the workspace area. With local path planning robot may avoid obstacles while moving tovards target.

Global Path Planning is a map-oriented deliberative system with relatively slower response. Global path planning assumes complete knowledge of the workspace area and obtains a feasible path leading to the goal.

Local navigation follows the *global path* and innitiates the next motion command based on local observations in real time. Local navigation generates a new path by overwriting the original global path in response to regional change in environment such as new obstacles.

[4] Approached probabilistically, the localization problem is a density estimation problem, where a robot seeks to estimate a posterior distribution over the space of its poses conditioned on the available data.

Chapter II

PATH PLANNING ALGORITHMS FOR ROS

PATH PLANNING ALGORITHMS FOR THE ROBOT OPERATING SYSTEM

The open-source Robot Operating System (ROS) [10][1] is a varied and scalable P2P network-based robotics framework. We present path-planning algorithms over a P2P network for collision-free autonomous ground-robot navigation under uncertainty constraints.

OPTIMALITY IN ROBOT MOTION: OPTIMAL VS OPTIMIZED MOTION

Motion planing is all about reaching given target and computing target if target exists. Optimal robot motion planning and control refers to finding a solution that optimize given criterion. Optimal robot motions, if they exist, provide solutions to find a path considering obstacles or criteria to be optimized. When we are not able to find an optimal solution for theoretical reasons [20], the problem needs to be reformulated either considering a discrete representation of space and/or time by slightly changing the optimization criterion, or by resorting to numerical optimization algorithms.

[1] Contributions are coordinated via ROS.org, compiling wiki documentation, and the ROS standards [11].

PEER-TO-PEER NETWORK

A P2P network is a decentralized and distributive network architecture in which all individual nodes are termed peers. In decentralized P2P networks, peers arrange themselves into an *overlay network*, a logical network overlaying a physical network. Nodes (*Vertices* in the ROS computational graph) are processes performing computation in the ROS system. Each *instance* must have a unique name and type (filesystem location of the executables). All messages are *packets* of data sent between ROS nodes. A communication link between ROS nodes is termed *topic.* ROS is a distributed computing environment compromising pottentially hundreds of nodes and multiple machines. Any node may communicate with any other node at any time. A ROS P2P network is loosely coupled and can use a few different methods of communication.

The following paragraphs describe *synchronous* RPC communication, *asynchronous* streaming, and the parameter server.

Synchronous RPC *communication over services:* The RPC protocol allows the construction of client-server applications, using a demand/response protocol with management of transactions. The client is blocked until a response is returned from the server, or a user-defined optional timeout occurs. The RPC guarantees at-most-once semantics for the delivery of the request. The RPC guarantees that the response received by a client is definitely that of the server and corresponds effectively to the request (and not to a former request to which the response might have been lost). RPC also allows a client to be unblocked (with an error result) if the server is unreachable or if the server had crashed before emitting a response. Additionally, this protocol supports the propagation of abortion through the RPC. This mechanism is termed *abort propagation*. When a thread that is waiting for an RPC reply is aborted, this event is propagated to the thread that is currently servicing the client request.

Asynchronous streaming of data over topics: *Asynchronous transmission* uses start and stop bits to signify the beginning and ending bits, so a character would actually be transmitted using ten bits instead of 8. For example, "`0101 1001`" would become "**`1`** `0101 1001` **`0`**". The extra one (or zero, depending on the parity bit) at the start and end of the transmission tells the receiver first that a character is coming and second that the character has ended. This method of communication is used when data are sent occasionally as opposed to in a solid stream. The start and stop bits in prevous example are in bold. The start and stop bits must be complementary. This allows the receiver to recognize when the second packet of information is being sent. Storage of data on the *parameter server*: A parameter server is a shared dictionary that is accessible via network APIs. Nodes use this server to store and retrieve parameters at runtime. As asynchronous transmission is not designed for high-performance, it is best used for static, non-binary data such as configuration parameters. That meant to be globally viewable so that tools can easily inspect the configuration state of the system and modify when necessary. The parameter server runs inside of the ROS master, which means that API is accessible via normal RPC libraries [1].

Developing *intelligent* robots that can accomplish production tasks without human help has been an importnant goal in artificial intelligence. From a technical point of view, autonomous robot navigation focuses primarily on generating optimal global paths to maneuver the robot to a target production station in a real time environment. Autonomous robots can reactively correct their course by circumventing obstacles with collision avoidance, and at the same time explore and map the unmapped region. When the robot senses remote target, the robot may not initially determine the distance accurately. Using parameter estimation and a Kalman Filter [5], the robot can estimate the target position most accurately. Using all the data the sensors have processed, the robot uses a Kalman Filter for parameter estimation and state estimation. Figure 1 illustrates measurement uncertainty associated with the estimation.

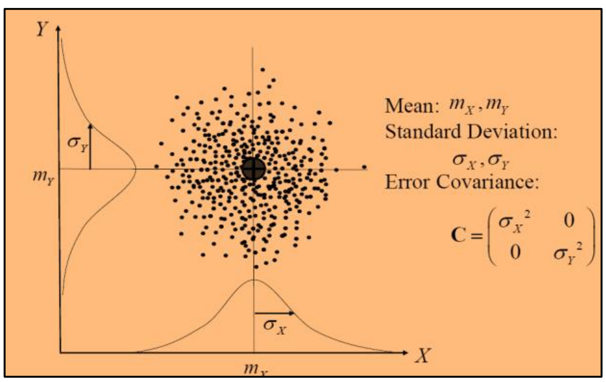

Figure 1: Quantifying Uncertainty [11]

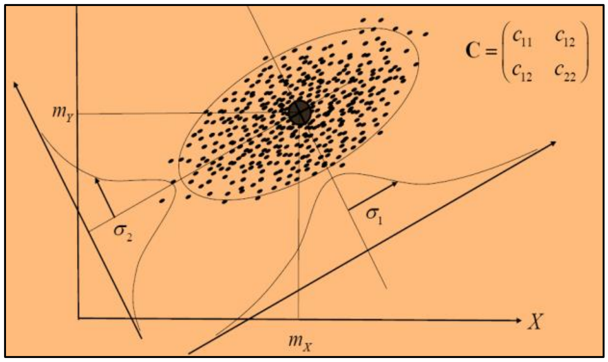

Figure 2: Covariance Matrix [11]

Figure 2 illustrates a covariance matrix which contains off-diagonal elements, reflecting correlation between two axes. In probability theory and statistics, a covariance matrix (also known as dispersion matrix or variance–covariance matrix) is a matrix whose element in the *i*, *j* position is the covariance between the *i* th and *j* th elements of a random vector (that is, of a vector of random variables). Each element of the vector is a scalar random variable, either with a finite number of observed empirical values or with a finite or infinite number of potential values specified by a theoretical joint probability distribution of all the random variables. Intuitively, the covariance matrix generalizes the notion of variance to multiple dimensions. As an example, the variation in a collection of random points in two-dimensional space cannot be characterized fully by a single number, nor would the variances in the *x* and *y* directions contain all of the necessary information; a 2×2 matrix would be necessary to fully characterize the two-dimensional variation.

Global path planning (GPP) addresses autonomous robot navigation in contexts including unmanned ground vehicles control [12], an unmanned aircraft [3], and the Mars Rover [13]. A *robot global path planning* (RGPP) system senses the information from the environment and plans a collision-free trajectory to navigate to a destination- in our case, the production station. Initially, GPP algorithms assumed a robot has complete knowledge of its environment, its production floor, and its own placement. However, in real applications, information is either partially available or completely unavailable in advance. Therefore, more recent work has focused on how to generate a global path in the presence of sensor noise and map incompleteness [7]. For GPP we used Dijkstra algorithm expressed in pseudo code, Figure 3 [15].

```
Function Dijkstra (Graph, source):
     for each vertex v in Graph:          // Initializations
         dist[v]:= infinity;              // Unknown distance function from
                                          // source to v
         previous[v]:= undefined;         // Previous node in optimal path
     end for                              // from source

     dist [source]:= 0;                   // Distance from source to source
     Q: = the set of all nodes in Graph;  // All nodes in the graph are
                                          // unoptimized - thus are in Q
     while Q is not empty:                // the main loop
         u: = vertex in Q with smallest distance in dist []; // Source node in first case
         remove u from Q ;
         if dist[u] = infinity:
             break;                        // all remaining vertices are
         end if                            // inaccessible from source

         for each neighbor v of u:         // where v has not yet been
                                           // removed from Q.
             alt: = dist[u] + dist_between (u, v);
             if alt < dist[v]:              // Relax (u, v, a)
                 dist[v]:= alt;
                 previous[v]:= u;
                 decrease-key v in Q;       // Reorder v in the Queue
             end if
         end for
     end while
     return dist;
 end function
```

Figure 3: Dijkstra's Pseudo Code [15]

Figure 5 shows the A* algorithm which combines both the Dijkstra and the Best First Search algorithms presented in Figure 6, to find the shortest path.The map navigation use ROS's `move_base` planner architecture. The default global planner

TrajectoryPlannerROS is a wrapper around Dijkstra's algorithm. It operates via dynamic window.[2]

```
1 OPEN = [initial state]
2 CLOSED = []
3 While OPEN is not empty
4 Do
         Remove the best node from OPEN, call it n, add it to CLOSED.
         If n is the goal state, backtrack path to n (through recorded parents)
         and return path.
         Create n's successors.
         For each successor do:
          a. If it is not in CLOSED and it is not in OPEN: evaluate it, add it to
                  OPEN, and record its parent.
          b. Otherwise, if this new path is better than previous one, change its
                  recorded  parent.
                  i.  If it is not in OPEN add it to OPEN.
                  ii. Otherwise, adjust its priority in OPEN using this new
                      evaluation.
5 Done
```

Figure 4: Best First Search Algorithm [15]

[2] A dynamic programming algorithm will examine the previously solved subproblems and will combine their solutions to give the best solution for the given problem.

```
Function A*(start, goal)
   closedset: = the empty set    // the set of nodes already evaluated.
   openset: = {start}      // the set of tentative nodes to be evaluated,
                           //initially containing the start node
   came_from:= the empty map    // the map of navigated nodes.
   g_score [start]:= 0    // Cost from start along best known path.
   // Estimated total cost from start to goal through y.
   f_score [start]:= g_score [start] + heuristic_cost_estimate (start, goal)
    while openset is not empty
       current: = the node in openset having the lowest f_score [] value
        if current = goal
          return reconstruct_path (came_from, goal)
          remove current from openset
          add current to closedset
        for each neighbor in neighbor_nodes (current)
          if neighbor in closedset
            continue
          tentative_g_score:= g_score [current]+ dist_between (current,neighbor)
          if neighbor not in openset or tentative_g_score < g_score [neighbor]
              came_from [neighbor]:= current
                g_score [neighbor]:= tentative_g_score
                f_score [neighbor]:= g_score [neighbor]+ heuristic_cost_estimate
                                    (neighbor, goal)
          if neighbor not in openset
                 add neighbor to openset
                 return failure
   Function reconstruct_path (came_from, current_node)
   if current_node in came_from
        p: = reconstruct_path (came_from, came_from[current_node])
        return (p + current_node)
    else
        return current_node
```

Figure 5: A* Pseudo Code [15]

Conventional Roadmap Methods for finding *shortest paths* include visibility graphs, which connect the nodes of polygonal obstacles, and Voronoi roadmaps which use borders as edges [2].Other methods in use are Probabilistic Roadmap Methods (PRM) (see Figure 6), Potential Field Methods (PFM), and Harmonic Potential Field Map Methods (HPFM).

```
INITIALIZE()
for all (cell ∈ cells) do
    cell.dist = distPointQueryLine(cell.origin);
end for
OP EN = {parentCell(n_init )}; 6:  CLOSED = ∅;
GROWPRMINCELL(cell)
numSamples = 0;
while (numSamples < nodeIncrementPerCell) do
        sample random configuration c_new in cell;
    cell.numTrials++;
    if (isFreeConfig(c_new)) then
          add node n_new  to N ;
          connect n_new  to neighbors, add edges to E;
          update cell.numComponents of cell;
          numSamples++;
    end if
end while
cell.numSamples += numSamples; 20:  updateOccupancy(cell);
updateConnectedness(cell);
updateValue(cell);
SOLVEQUERY(n_init ,n_goal ) 24:  Initialize();
add n_init , n_goal  to N ;
connect n_init , n_goal  to neighbors, add edges to E; 27:  loop
    if (OP EN = ∅) then
        report failure;
    end if
    cell = takeFirst(OP EN );
    GrowPRMInCell(cell);
    PerformRandomWalksInCell(cell);
    if (cell.occupancy > occupancyThresh or cell.numSamples ≥
                           maxNodesPerCell) then
          CLOSED ← cell;
    else
          OP EN ← cell;
    end if
    for all (neighbor ∈ neighbors(cell)) do
          if (neighbor ∉ CLOSED) then
                OP EN ← neighbor;
    end if
    end for
    sortAscendingByValue(OP EN );
    if (parentComp(n_init ) = parentComp(n_goal)) then
          path = findShortestPath(G);
          if (path satisfies quality condition) then
                return  path;
          else
                publish path;
          end if
    end if
end loop
MAIN()
create array cells; 56:  N = ∅;
E = ∅;
G = (N, E);
for all (query ∈ queries) do
     solutionPath = SolveQuery(query.n_init,query.n_goal);
```

Figure 6: Cell Probabilistic RoadMap Method (PRM) Pseudo Code [17]

The current position estimate of the robot is given by calculating the mean of the previous paths. However, in practice, the estimate is likely to be noisy and we have to take this uncertainty into account in order to ensure collision free and efficient paths. The CPP is performed by robots in fully observable environments. System conditions are usually assumed to be deterministic and discrete. Probability based approaches aided by statistical tools (extended Kalman Filter (EKF) [5] and particle filter–sequential Monte Carlo [6]) were employed. The Extended Kalman Filter, Figure 7. The EKF has been considered the standard in the theory of nonlinear state estimation, navigation systems, and GPS.

	Standard Kalman Filter	Extended Kalman Filter
	Linear systems	Nonlinear systems
State Equation	$x_{t+1} = A_t x_t + B_t u_t + G_t w_t$	$x_{t+1} = f(x_t, u_t, t) + G_t w_t$
Output Equation	$y_t = H_t x_t + v_t$	$y_t = h(x_t, t) + v_t$

Figure 7: Extended Kalman Filter (EKF) Deals with Nonlinear Systems Represented by Nonlinear State and Output Equations [5], [11]

Suppose that robot is at point *A* at the time *t* having a state estimation error covariance P_t. The robot needs to acquire more data to better estimate the location of its position relative to the target.

Gathering of new data provides the robot with more useful information, see Figure 8.

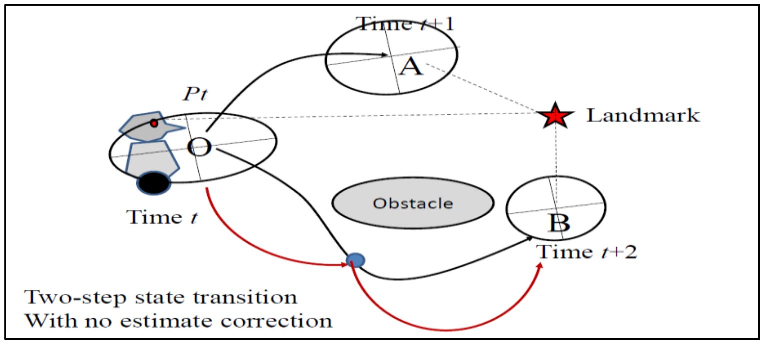

Figure 8: Two-step State Transition with No Estimate Correction [11]

The *particle filters* or *sequential Monte Carlo* (SMC) [6] method consists of a set of on-line posterior density estimation algorithms that estimate the posterior density of the state-space by directly implementing the Bayesian recursion equations.[3] SMC methods use a grid-based approach, and use a set of particles to represent the posterior density. SMC methods provide a methodology for generating samples from the required distribution without requiring assumptions about the state-space model or the state distributions [6]. Figure 9 generates a map of the environment; the goal of this algorithm is for the robot to determine its position within the environment.

```
Function MCL:     ( X_t-1,  u_t, z_t)
        X_t = X_t = 0
                 for m = 1 to N
                          X_t^(m) = motion_update(u_t, x_t-1^(n))
                          W_t^(m) = sensor_update(z_t, x_t-1^(n))
                          X_t = X_t +{X_t^(n), w_t^(n)}
                 end for
                 for m  = 1 to M
                 draw x_t^[i] from X̄_t with probability ∝ w_t^[i]
                          X_t = X_t+ X_t^(i)
                 end for
        return X_t
```

Figure 9: Particle Filter-sequential Monte Carlo Algorithm [18]

[3] The on-line algorithm receives one example at the time and maintains a parameter that is essentially an average of the past examples [20].

The basic MCL algorithm, in the next set of samples S_{t+1} from current set S_t is illustrated in Figure 10. The next x_t is the location and the w_t are the probabilities (x_t, wt) pair represents the sample. The distance traveled is u_t and the sensor reading is z_t. . The location of sample i at the time t is $x_t^{(i)}$, where n is number of samples.

```
Inputs: Distance u_t, sensor reading z_t, sample set S_t ={(x_t^(i), wt^(i) )| i= 1…n}
      for  i=1 to n do                    // first update of current set of samples
              x_t = updateDist(x_t,u_t)    // compute new location
              w_t^(i) =prob(z_t|x_t^(i))      // compute new probability
      S_t+1 = null                         // resample for next generation of samples
      for I = 1 to n do
              Sample an index j from distribution given in weights S_t
      Add (x_t^(i), wt^(i)) to S_t+1// Add sample j to new set of samples
      return S_t+1
```

Figure 10: MCL Basic Algorithm [18]

SIMULTANEOUS LOCALIZATION AND MAPPING (SLAM)

During the process of path planning, the robot is continuously learning its environment. By building a map of the environment where the robot is locating and localizing itself concurrently is termed Simultaneous Localization and Mapping (SLAM). SLAM shown in Figure 11, maintains probability distribution over *current* pose and map (step 1), then predict time (step 2), and finally update the measure (step 3).

1. $p\ (x_{t+1}, M \mid z^{t+1},\ u^{t+1}),$
2. $p\ (x_t, M \mid z^t, u^t)\ \ u_{t+1}\ \ \ \ p(x_{t+1}, M \mid z^t, u^{t+1}),$
3. $p\ (x_{t+1}, M \mid z^t, u^{t+1})\ \ z_{t+1}\ \ \ \ p(x_{t+1}, M \mid z^{t+1}, u^{t+1}),$

Figure 11: Probability Distribution over Current Pose

Local navigation follows the *global path* and determines the next motion command based on local observations in real time. Local navigation generates a new path by overwriting the original global path in response to a regional change in an environment such as new obstacles.

SPOTT's local path planner[4] is based on a potential field method using harmonic functions, which are guaranteed to have no spurious local minima [19]. A harmonic function on a domain $\Omega \in R^{n}$ is function which satisfies Laplace's equation:

$$V * V\emptyset =_{.} \sum_{n=1}^{\infty} \left(. \frac{n\delta * \delta\emptyset}{\delta * \delta * X * X} \right) = 0$$

The value of $\emptyset$ is given on a closed domain Ω in the configuration space ϵ.

The default local planner is termed *TrajectoryPlannerROS* which wraps an approach called *dynamic window*. The dynamic window algorithm is a velocity-based local planner that calculates the optimal collision-free velocity for a robot required to reach its goal. It translates a Cartesian goal (*x, y*) into a velocity (*v, w*) command for a mobile robot. There are two main goals, calculate a valid velocity search space, and select the optimal velocity. The search space is constructed from the set of velocities producing a safe trajectory, i.e., allow the robot to stop before colliding, given the set of velocities the robot can achieve in the next time slice its dynamic *dynamic window*. The optimal velocity is selected to maximize the robots clearance, maximize the velocity and obtain the heading closest to the goal as shown in Figure 12.

[4] A path planning framework has been proposed and implemented as part of the SPOTT mobile robot control architecture (Zelek 1996).

```
BEGIN DWA (robotPose, robotGoal,robotModel)
  desiredV = calculateV(robotPose,robotGoal)
  laserscan = readScanner()
  allowable_v = generateWindow(robotV, robotModel)
  allowable_w = generateWindow (robotW, robotModel)
  for each v in allowable_v
    for each w in allowable_w
    dist = find_dist(v,w,laserscan,robotModel)
    breakDist = calculateBreakingDistance(v)
    if (dist > breakDist)  //can stop in time
        heading = hDiff(robotPose,goalPose, v,w)
        clearance = (dist-breakDist)/(dmax - breakDist)
        cost = costFunction(heading,clearance, abs (desired_v - v))
       if (cost > optimal)
          best_v = v
            best_w = w
            optimal = cost
    set robot trajectory to best_v, best_w
  END
```

Figure 12: Local Planner Algorithm DWA

In summary, real-world robot navigation has significant challenges such as path planning, obstacle avoidance, fault isolation, and system resiliance. Autonomous robot movement (navigation) is a collection of hybrid design systems which are capable of handling all aspects of robotic navigation requirement. This paper presented paths algorithms used in the ROS. The open source library maintained by the robot volunteer community has proven a useful and helpful tool for all educational purposes. The *probabilistic* techniques are promising. Global planning, frameworks capable of coping with uncertainty have become increasingly popular. Partially Observable Markov Decision Process (POMDP) [4], and SLAM are advances in solving problems in global planning, local navigation, and exploration. The robot’s local movement has a number of methods proposed for obstacle avoidance and horizon control. Research in autonomous robot navigation has made significant progress. Navigation applications require the capability to solve problems offering nearly optimal solutions. For this reason, global planning algorithms, local navigation routines, and exploration procedures must be integrated to achieve a global goal.

Chapter III

COMPARISON: LENSED CAMERA VS LENSLESS CAMERA

For our research, we are using a 10 MP CMOS digital image sensor CMOS which has an active–pixel digital imaging sensor of 3856H x 2764V including border pixels. It can support both digital still images and a 1080p (3840H x 2160V) video digital mode. As a progressive–scan sensor it generates a stream of pixel data at a constant frame rate. If operated in 4:3 still-mode, the sensor generates 15 frames per second (fps), resulting in a full resolution image. An analog–to-digital converter (ADC) generates a 12 bit value for each pixel. The sensor uses a Bayer color pattern shown in Figure 13.

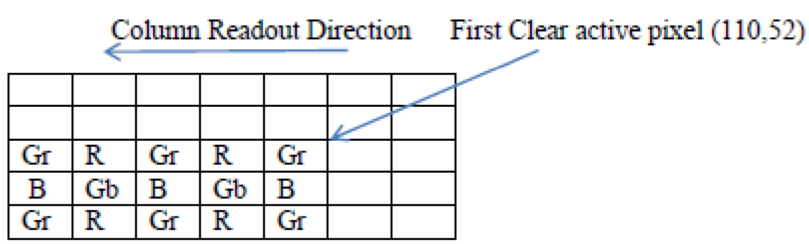

Figure 13: Pixel Color Pattern

The pixel array consists of even numbered rows containing green and red pixels, and odd-numbered rows containing blue and green pixels. Even-numbered columns contain green and blue pixels, and odd-numbered columns contain red and green pixels.

ORIGINAL SETTING USES LENSED CAMERA

We will compare an original image made with a lensed camera, presented in Figure 14 and Figure 15, with an initial original image from a lensed camera, given without any software or hardware adjustments.

In Chapter III we present graphic analysis of our original image in Figure 14 and graph analysis of the image from the lensed camera figure 14, obtained as a result of our improvements.

Figure 14: Original Image–Lensed Camera

In Figure 15 we present Analysis graph of the original image intensity.

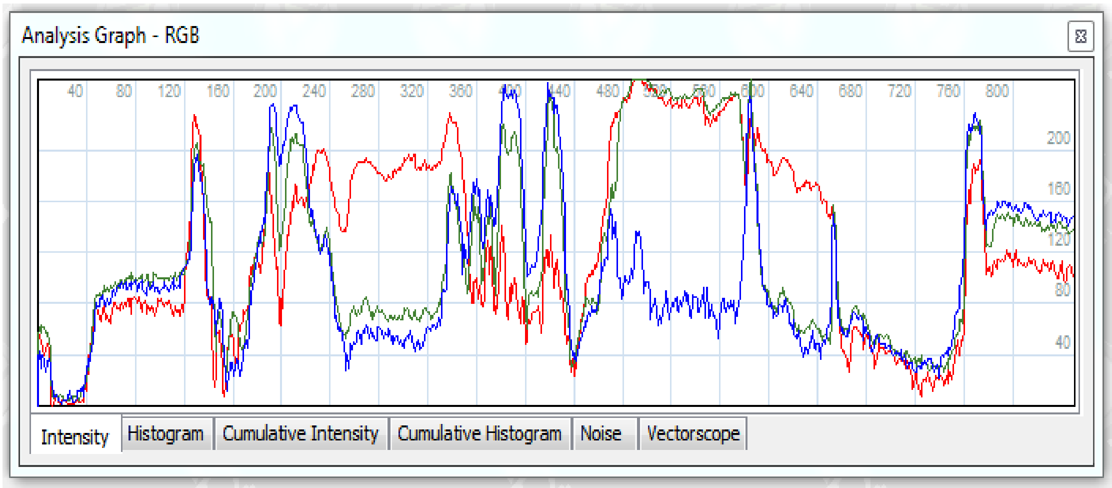

Figure 15: Analysis Graph–Original Image Intensity

In Figure 16 and Figure 18 we see histogram and cumulative histogram of the original image.

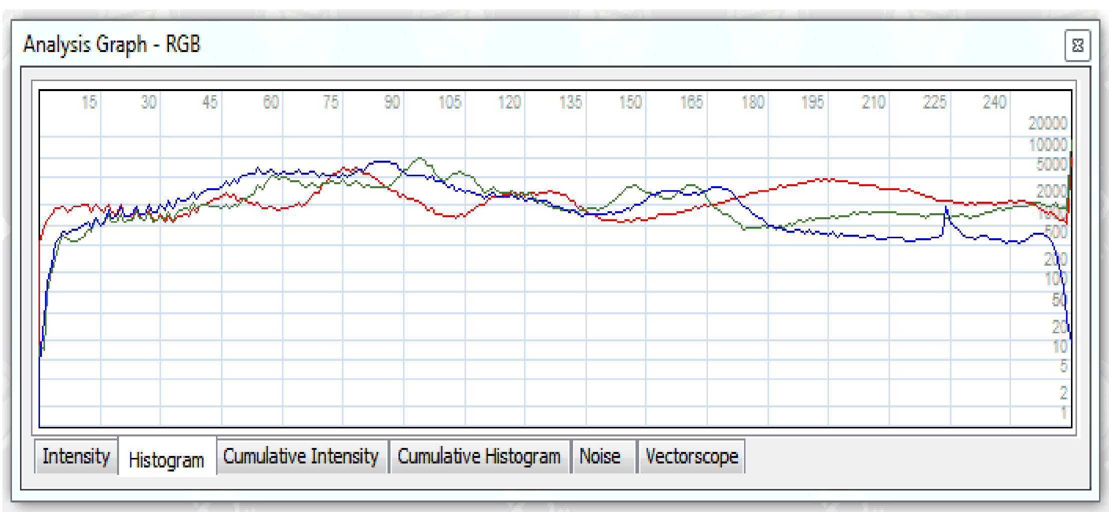

Figure 16: Analysis Graph–Original Image Histogram

Figure 17 shows Analysis cumulative intensity graph of Original image.

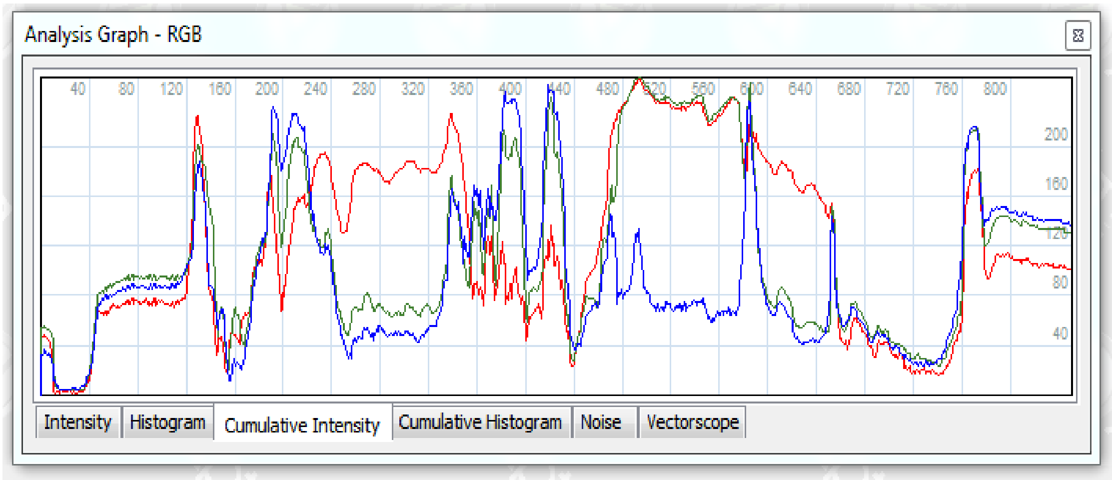

Figure 17: Analysis Graph–Original Image Cumulative Intensity

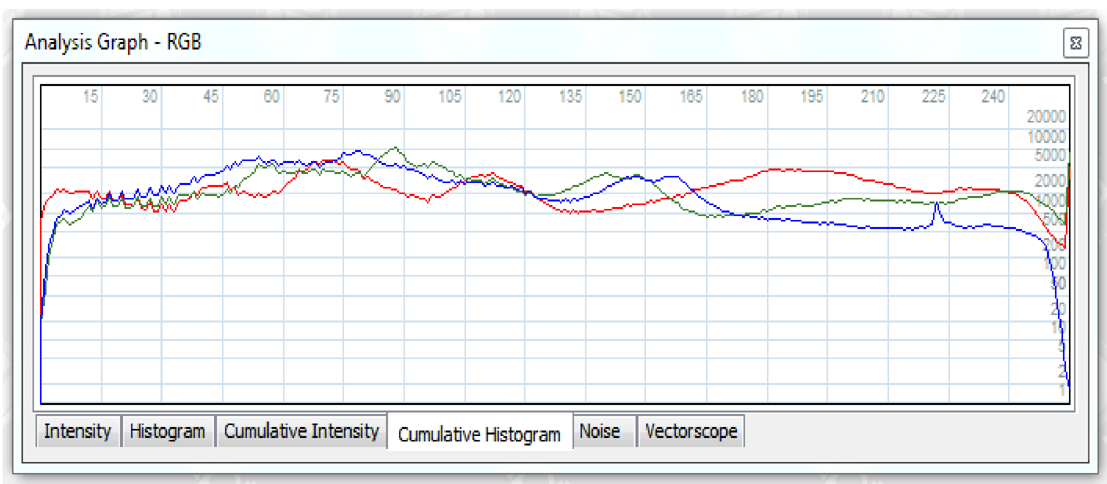

Figure 18: Analysis Graph–Original Image Cumulative Histogram

Noise analysis graph presented in Figure 19 shows uniformity.

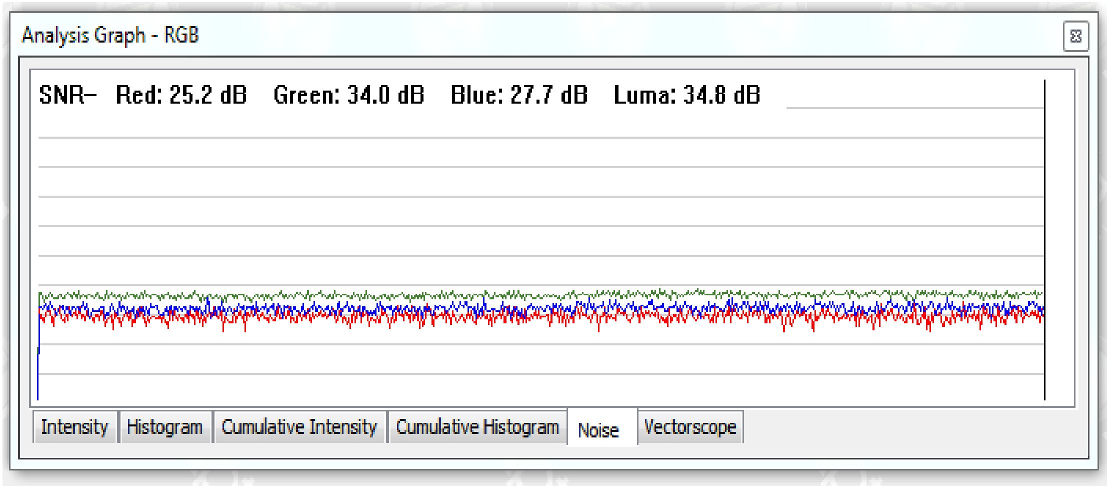

Figure 19: Analyses Graph–Original Image Noise

The uniform layout of the green dots in Figure 20 shows a vectorscope analysis graph of the original image.

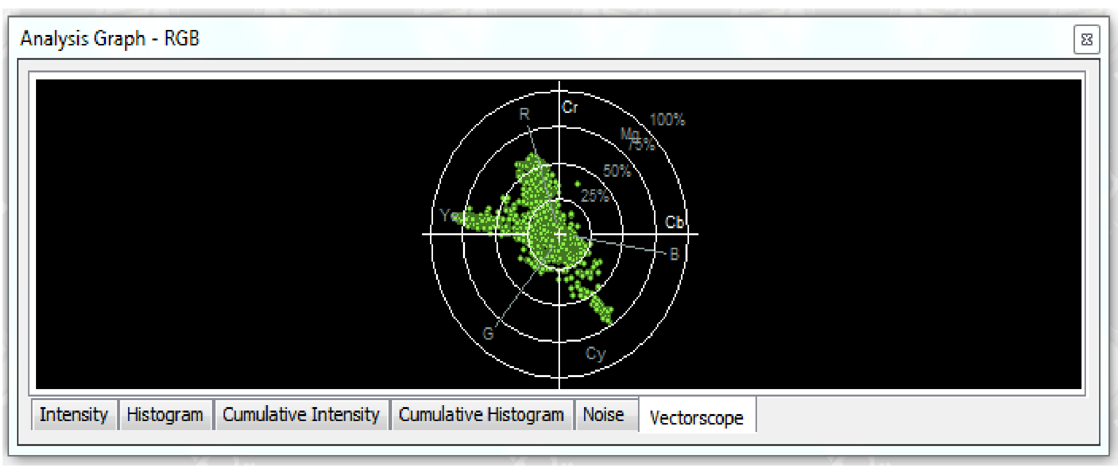

Figure 20: Analysis Graph–Original Image Vectorscope

The original image from the camera using a lens in Figure 21 presents focus in metric. Uniform green dots are spread over the image. In Figure 22 the focus metric graph shows the same linear uniformity.

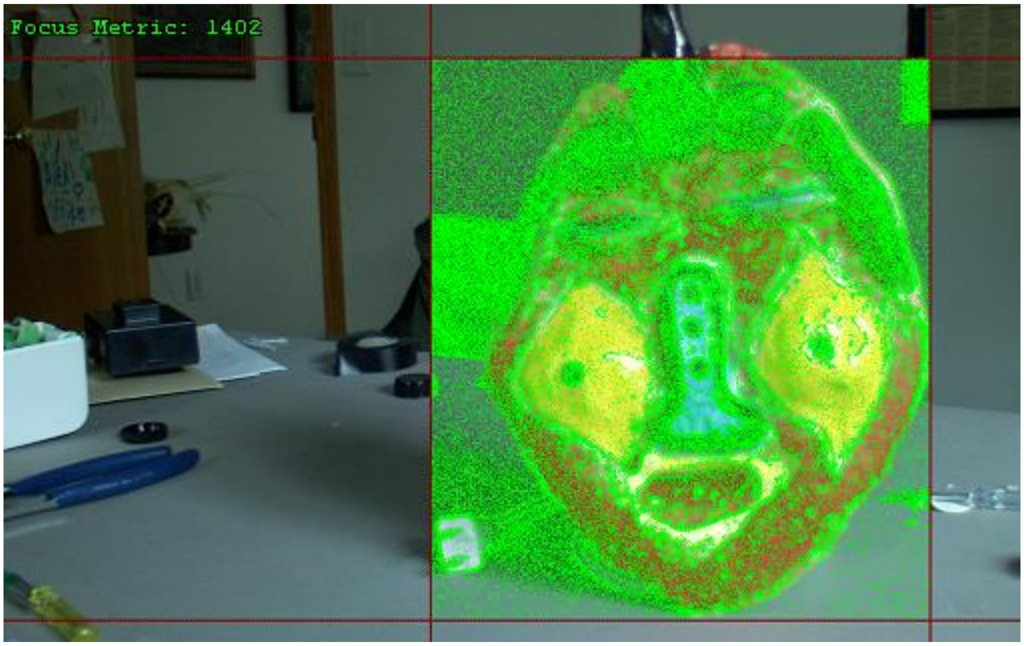

Figure 21: Original Image–Focus Metric

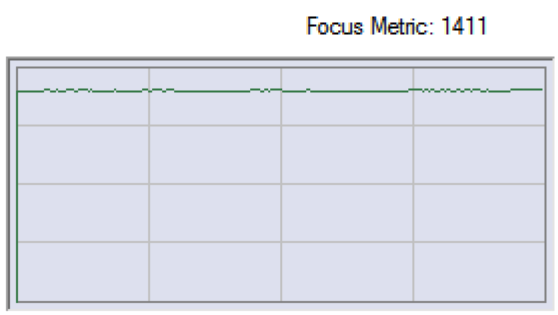

Figure 22: Original Image–Focus Metric Graph

For the original image we used the default settings from the camera and optical sensor manufacturer as we can see in Figure 23. Adapting Average-Near Algorithm, Flat Region Filter Algorithm, and Temporal Averaging algorithm are not applied. Aggressiveness is set to 2, and Temporal Depth was set to 2.

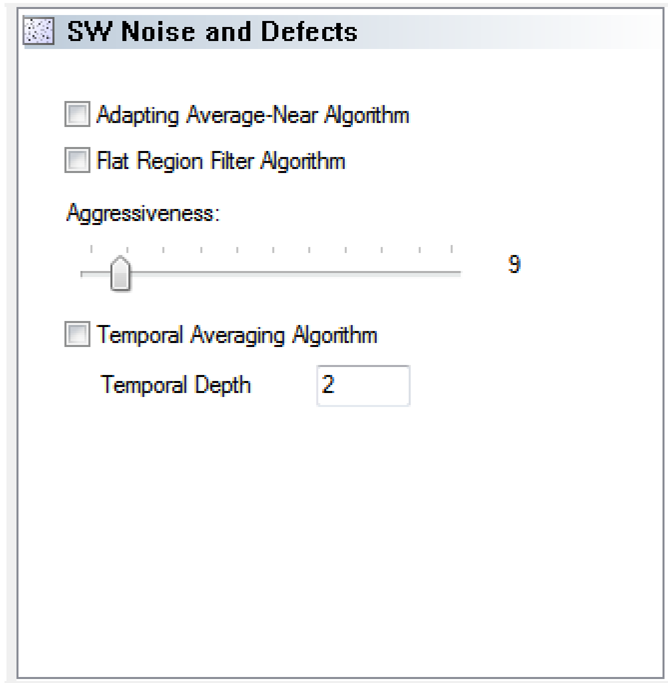

Figure 23: Original Image–Noise and Defects

LENSLESS CAMERA

During our design of the experiment, we went through two different hardware models. We will present hardware models of the lensless camera with holes of different diameters, set in near-object and far-object environments. We removed the lens from the optical sensor and applied our lensless camera instead.

Lensless Camera with Larger Apperture

Using a bigger apperture opening in a close object setting, we could not see much of the image. Edges are not very noticeable, but we can register movement. Figure 24 shows the image from the lensless camera in a near-object setting.

Figure 24: Image with Lensless Camera using larger apperture

Figure 25 shows a noise analysis graph of the lensless camera image. The separation between the green line and the blue and red lines is different from the graph of the original image shown in Figure 18.

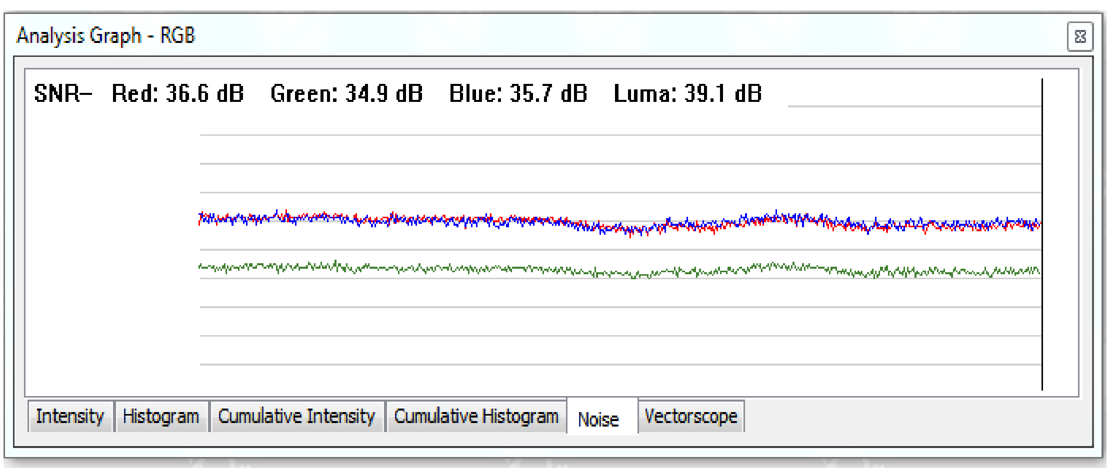

Figure 25: Analysis Graph–Lensless Camera Noise

The lensless camera image Vectorscope presented in Figure 26 shows only one green dot.

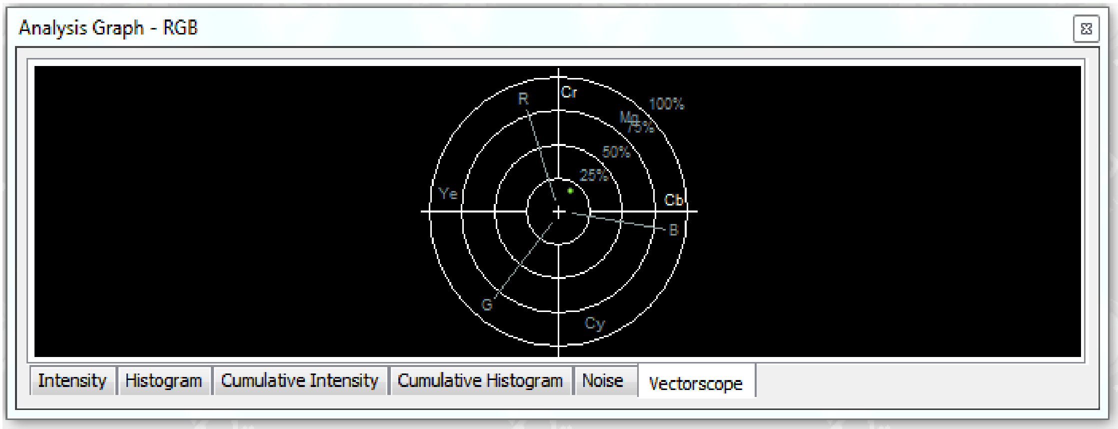

Figure 26: Analysis Graph–Original Image, Lensless Camera Vectorscope

Lensless Camera with Smaller Apperture

In this experiment we reduced the diameter of the apperture and applied it to the same optical sensor in near-object setting. After these changes, the edge of the image and its color dispersion can be seen in Figure 27.

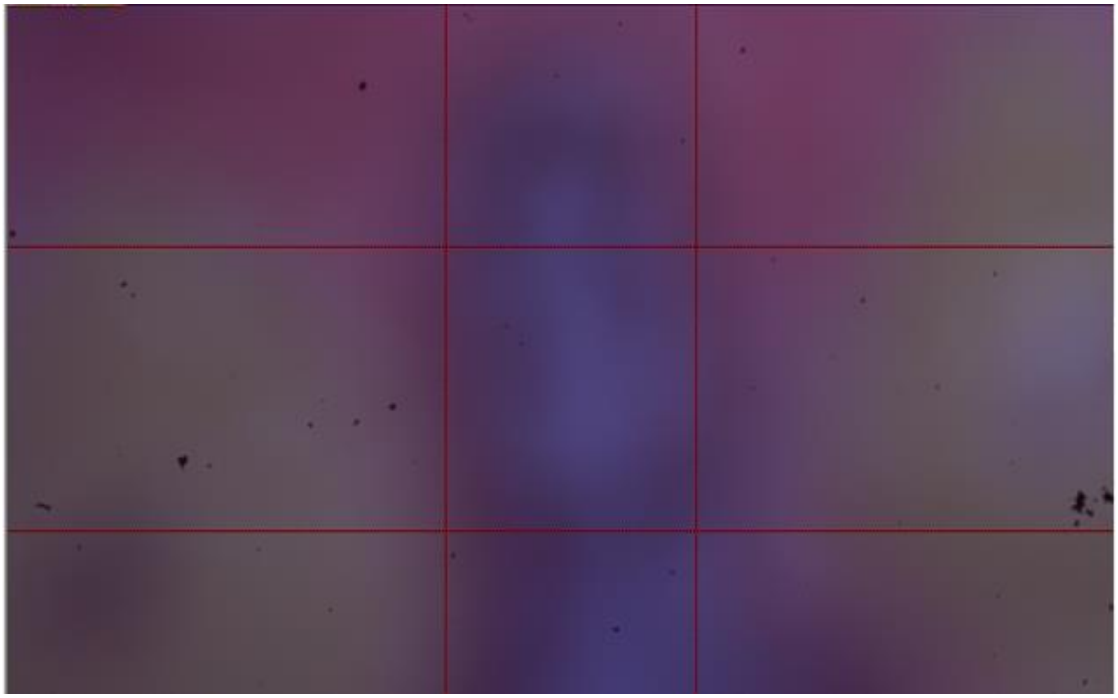

Figure 27: Image Made with Lensless Camera Using Small Apperture

Figure 28 shows image given with software algorithms application to the lensless camera original image In Figure 27. We reduced aggressiveness setting to 2 and increased temporal depth setting to 6 as shown in Figure 29.

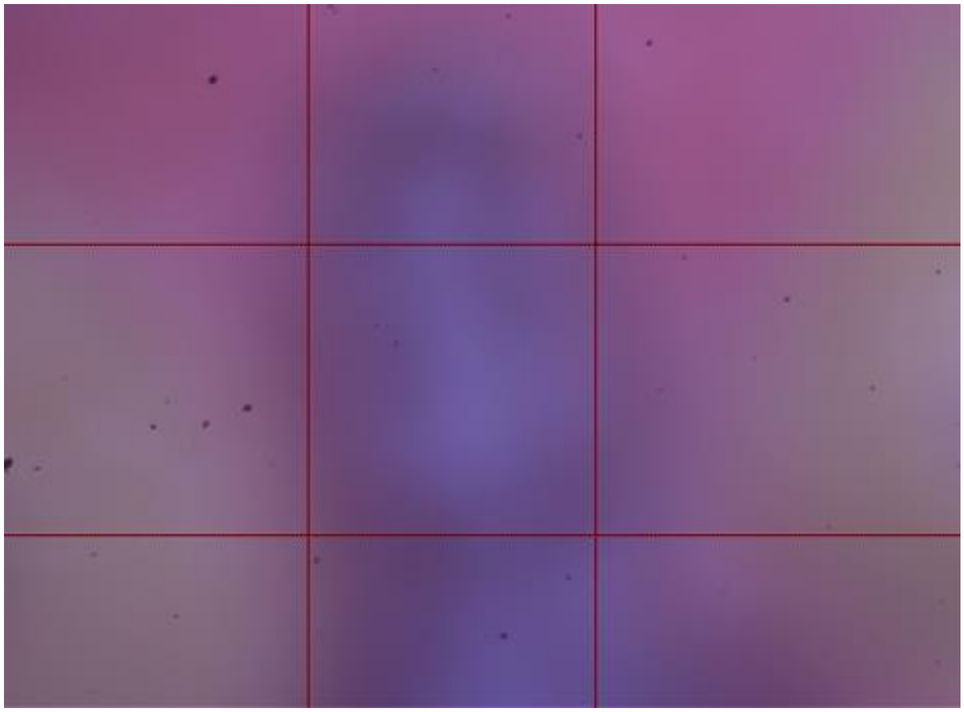

Figure 28: Image Made with Lensless Camera and Smaller Apperture

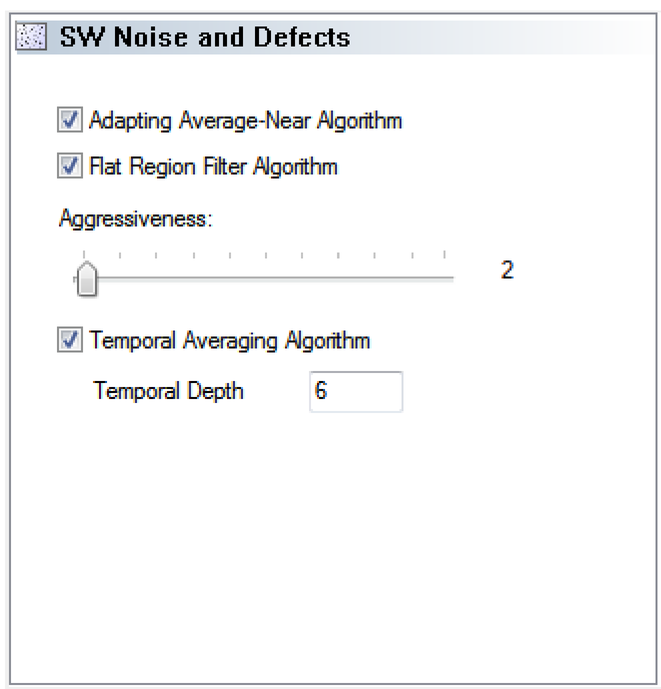

Figure 29: Changes Made with Lensless Camera–with Applied Algorithms

The lensless camera image analysis graphs shown in Figures 30-35 are different from the graphs presented in the previous paragraph Figures 15-20.

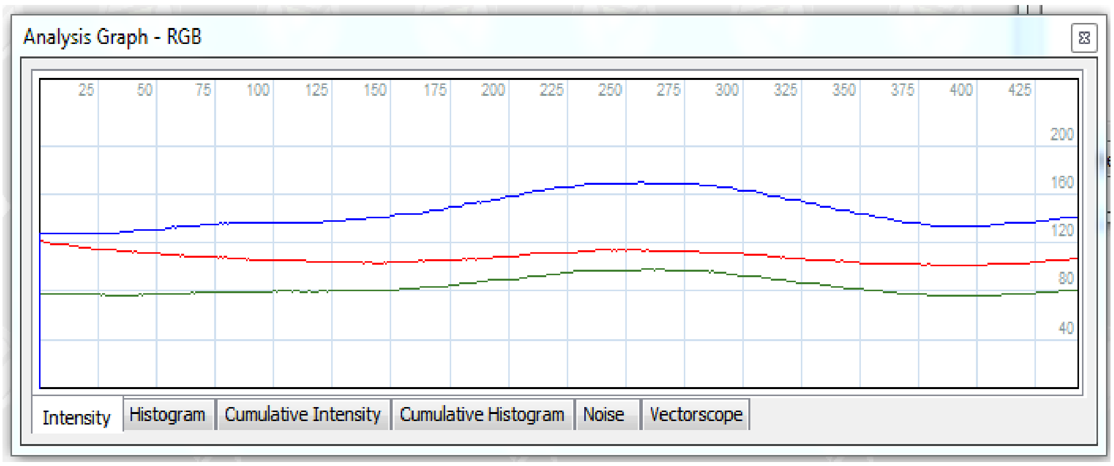

Figure 30: Analysis Graph Intensity Image Made with Lensless Camera

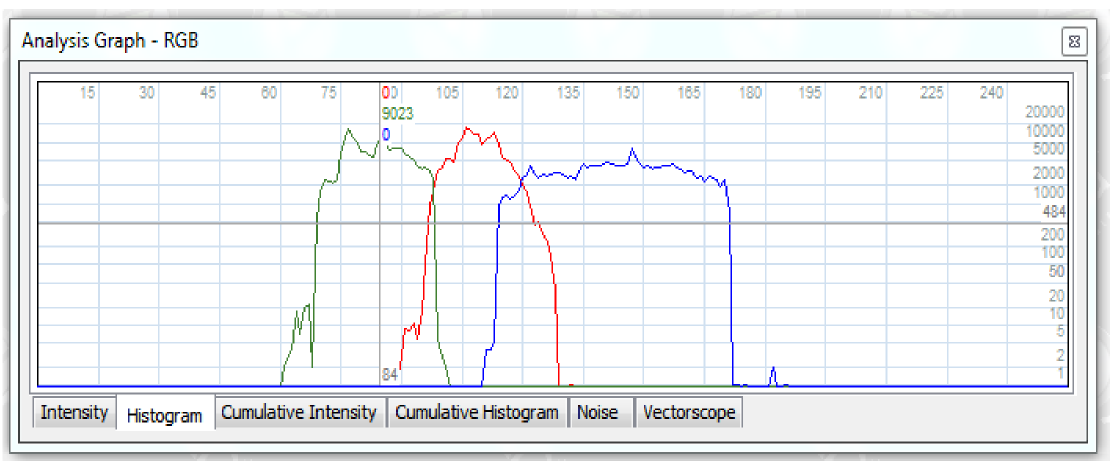

Figure 31: Analysis Graph Histogram Image Made with Lensless Camera

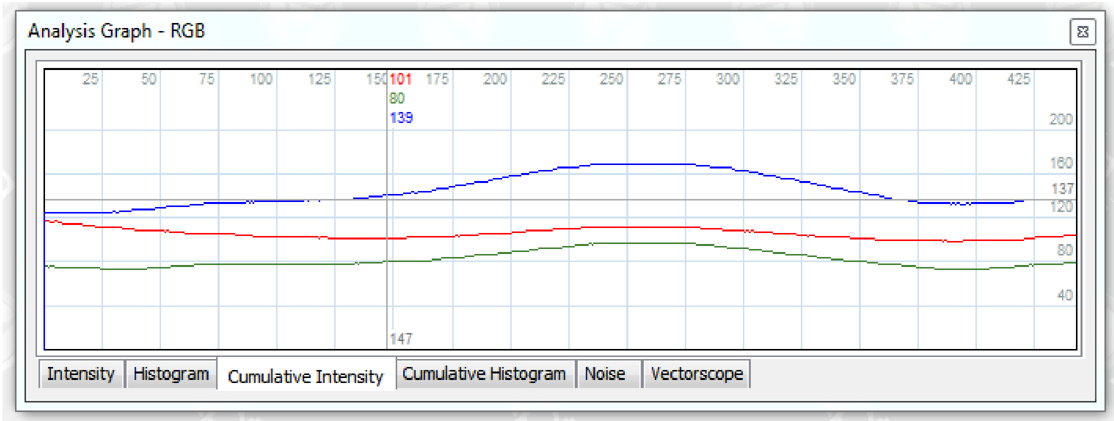

Figure 32: Analysis Graph Cumulative Intensity Image Made with Lensless Camera

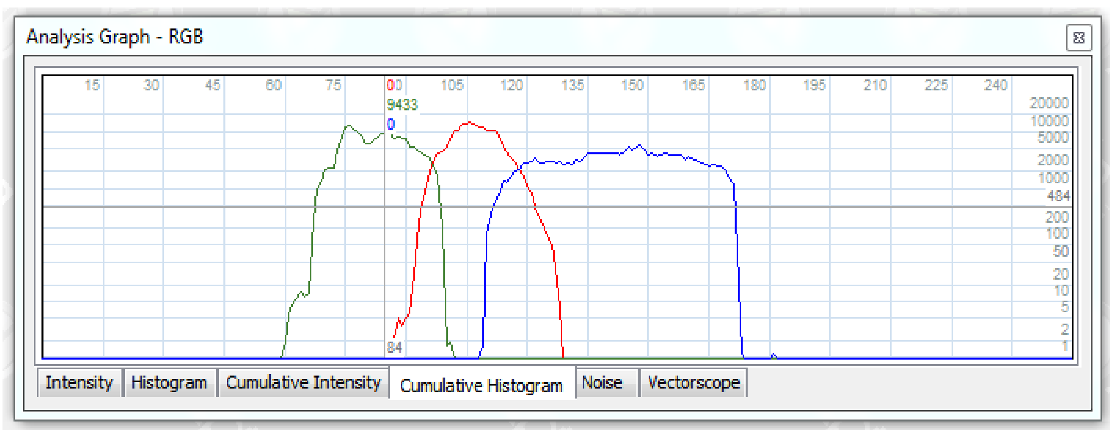

Figure 33: Analysis Graph Cumulative Histogram Image Made with Lensless Camera

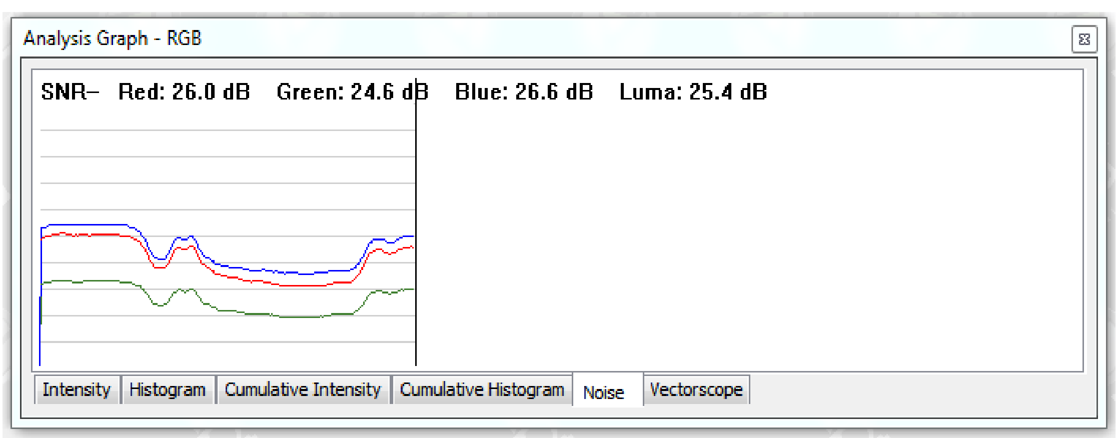

Figure 34: Analysis Graph Noise Image Made with Lensless Camera

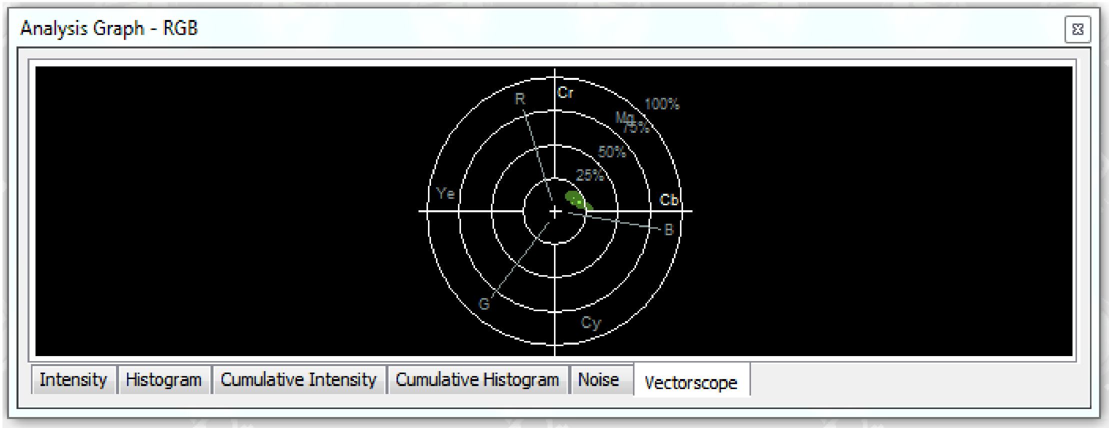

Figure 35: Analysis Graph Vectorscope Image Made with Lensless Camera

Lensless Camera with Small Diameter
Opening, and Increased Exposure

In this section we manually increased exposure to the lensless camera and magnified the image producing the image in Figure 36. Now we can see half of the face but clearly all edges in their respective colors.

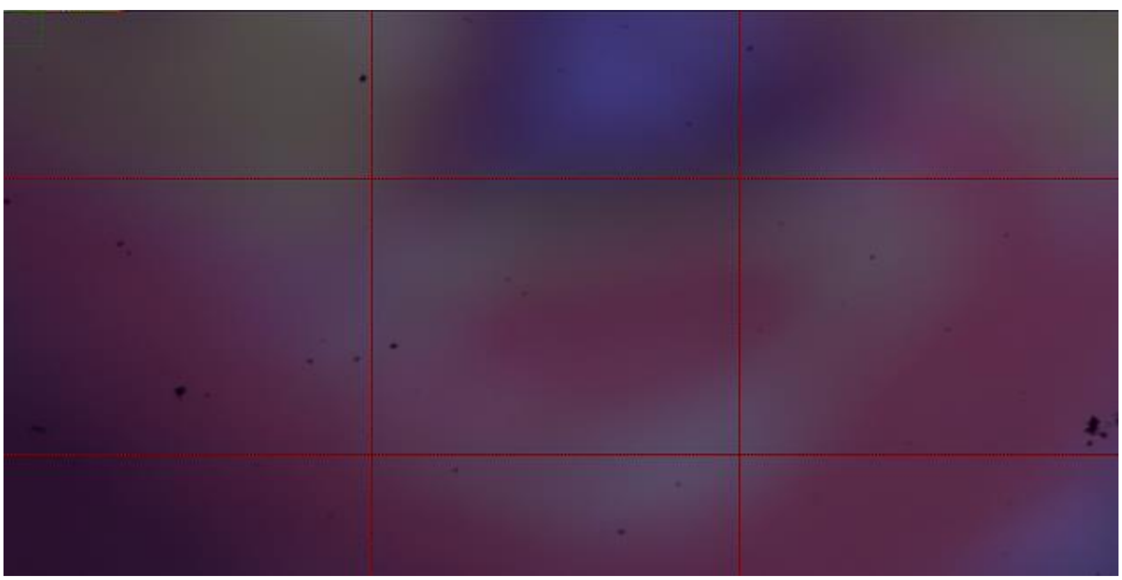

Figure 36: Image Made with Lensless Camera–Applied Manual Gain and High Exposure

Figure 37 presents manual exposure settings. We used a pixel integration time of 666.6 ms, with the absolute gain for red set at 1.00, green at 1.00, and blue at 1.56.

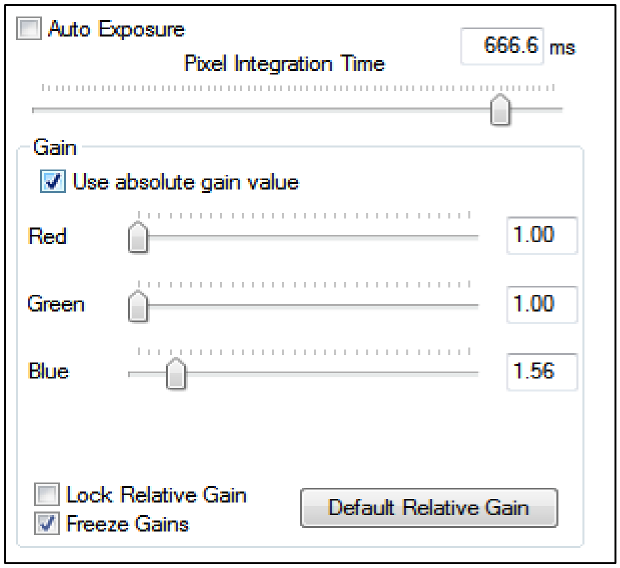

Figure 37: Gain and Exposure Parameters

Figure 38 presents a vectorscope analysis graph. Green dots are visible present and grouped in center.

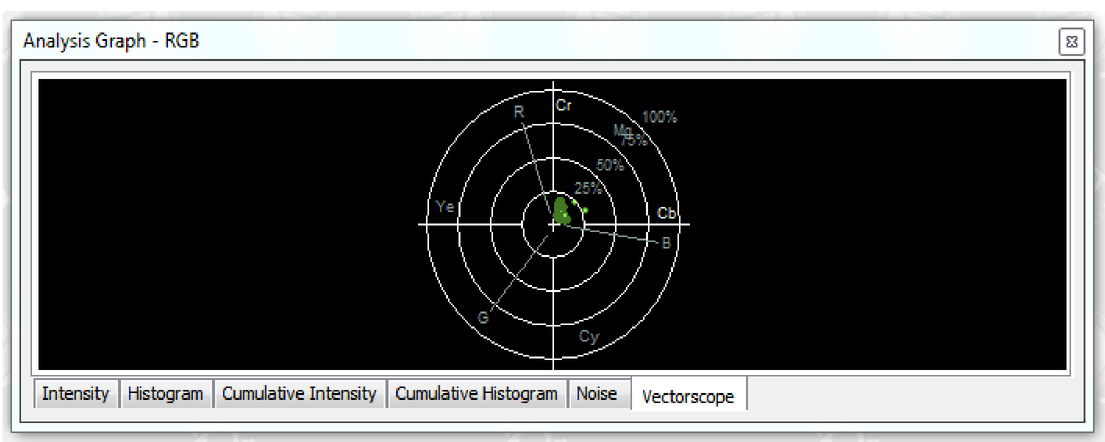

Figure 38: Analysis Graph of a Cumulative Vectorscope Made with Lensless Camera, Adjusted Gain and Exposure

Figure 39 shows focus metric 5.

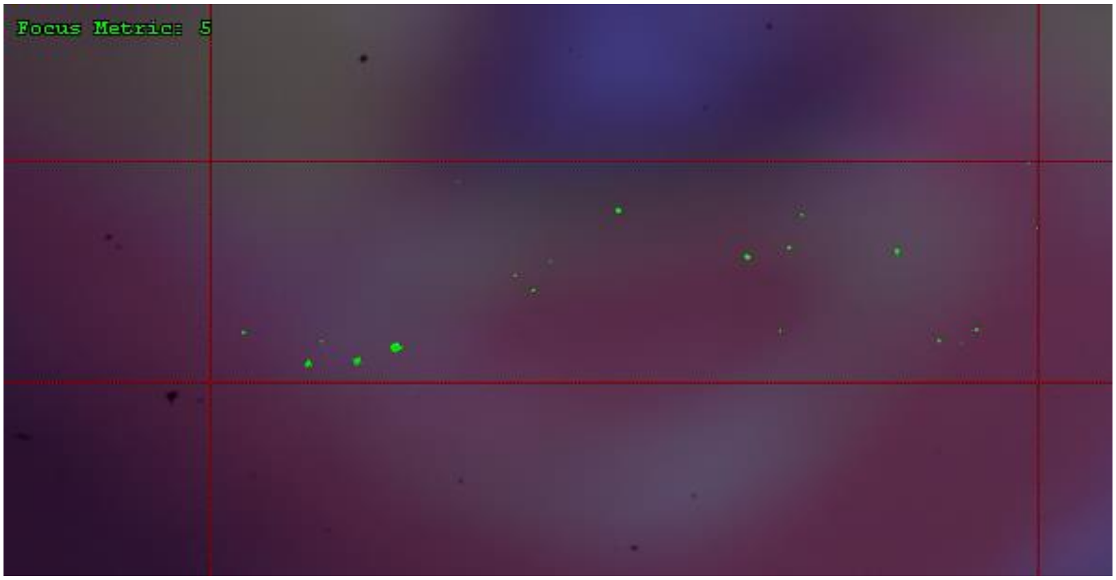

Figure 39: Image Made with Lensless Camera–Focus Metrics

Figure 40 presents a lensless camera image with software Bayer-bit unswizzling. We can clearly see color dissipation and face edges.

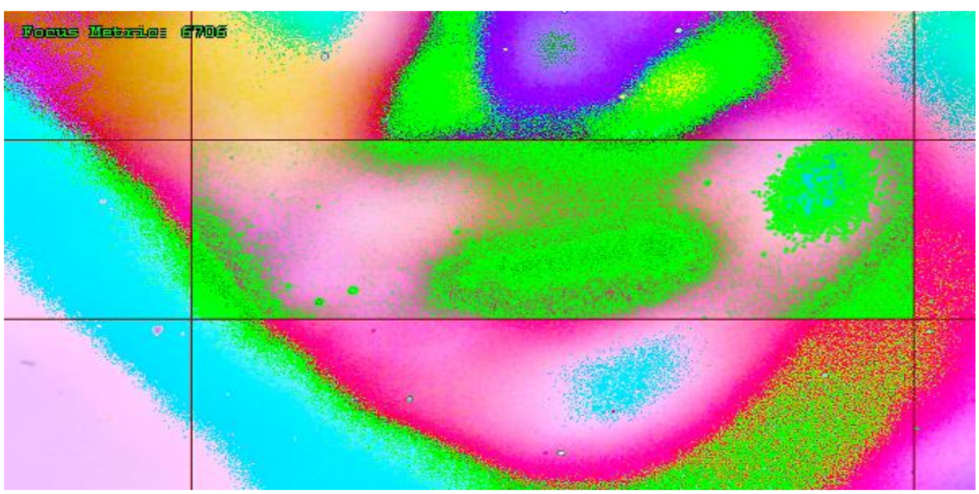

Figure 40: Image Made with Lensless Camera–Focus on Edges with Software Unswizzling

The data interpretation shown in Figure 41 presents our settings for the image obtained by the lensless camera.

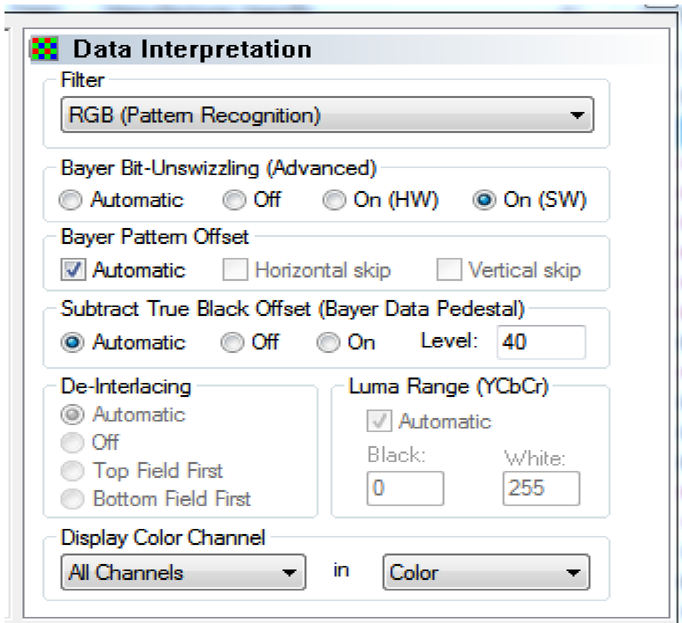

Figure 41: Data Interpretation

In Figure 42 we see image a cumulative intensity analysis graph for our lensless camera image.

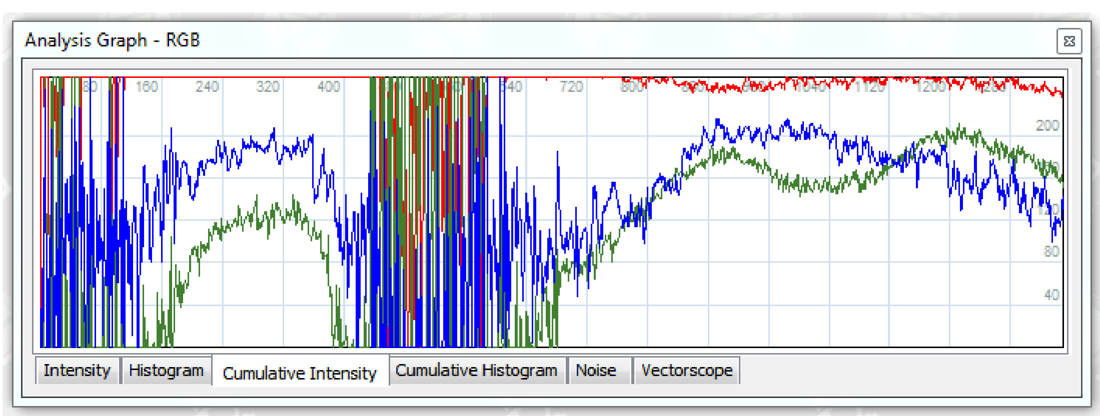

Figure 42: Analyses Graph Cumulative Intensity Image Made with the Lensless Camera

Cumulative histogram is presented in Figure 43.

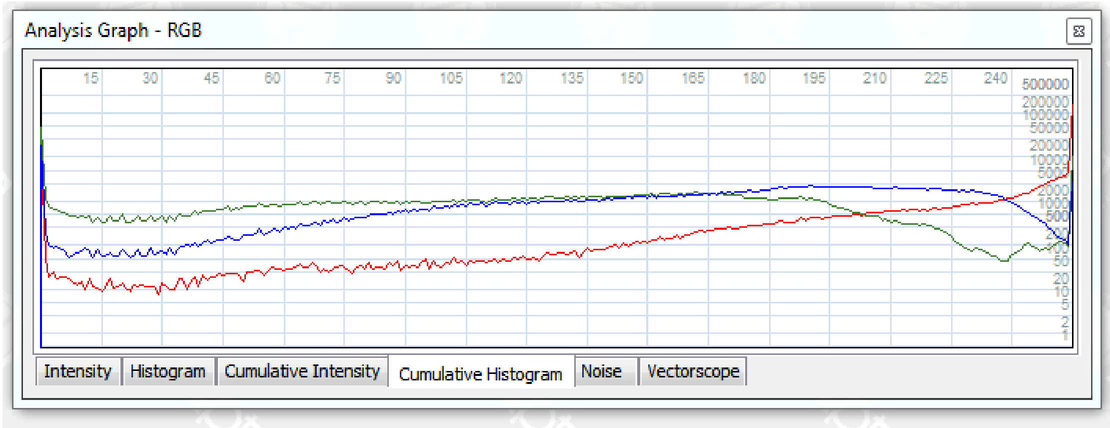

Figure 43: Cumulative Histogram Image Made with Lensless Camera

The vectorscope analysis graph in Figure 44 shows uniform green dot concentration.

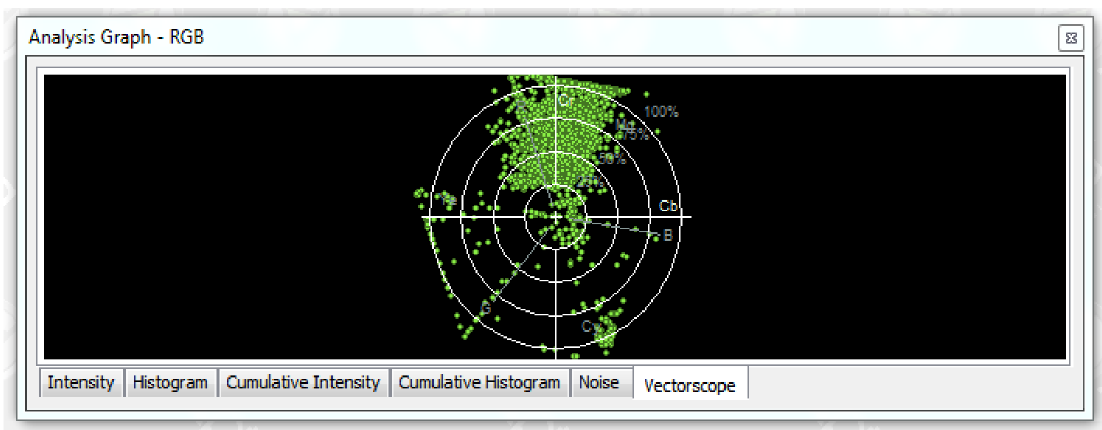

Figure 44: Cumulative Vectorscope Made with Lensless Camera

The tiled grayscale focused image presented in Figure 45 shows distinct face edges.

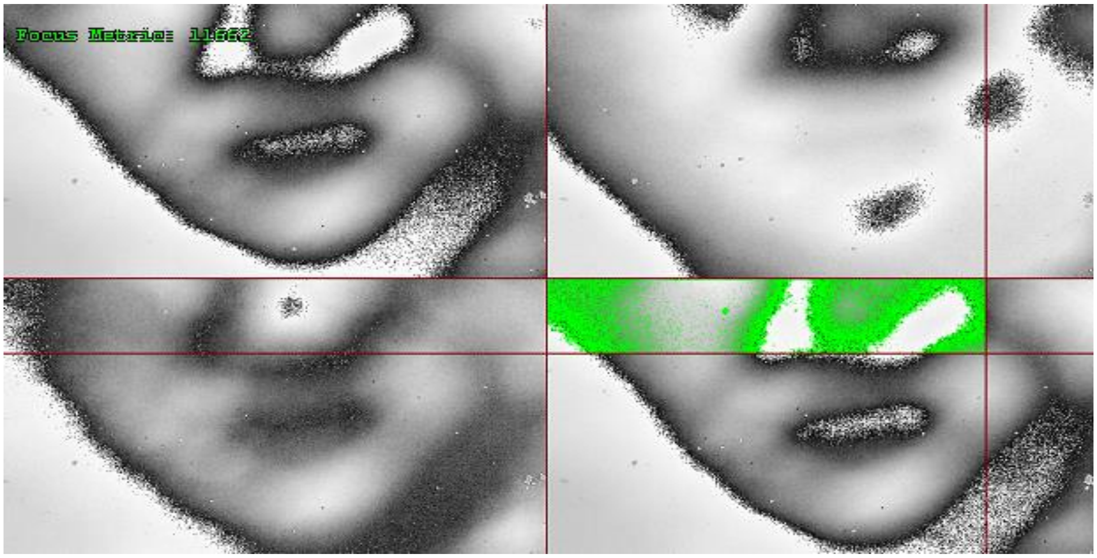

Figure 45: Image Made with Lensless Camera–Focused, Tiled Grayscale

In our implementation, we included the automatic Bayer pattern and subtract the true block offset. The parameters are shown in Figure 46.

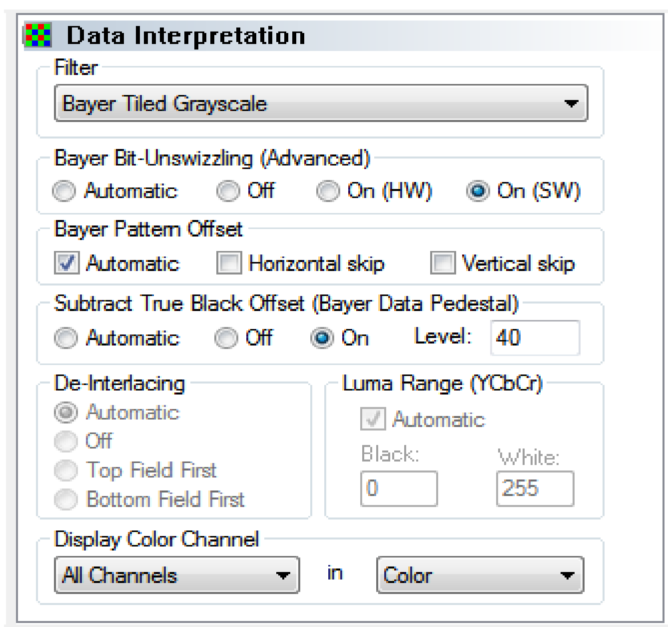

Figure 46: Data Interpretation

In Figure 47, we see a lensless camera tiled grayscale given with hardware unswizzling instead of software unswizzling.

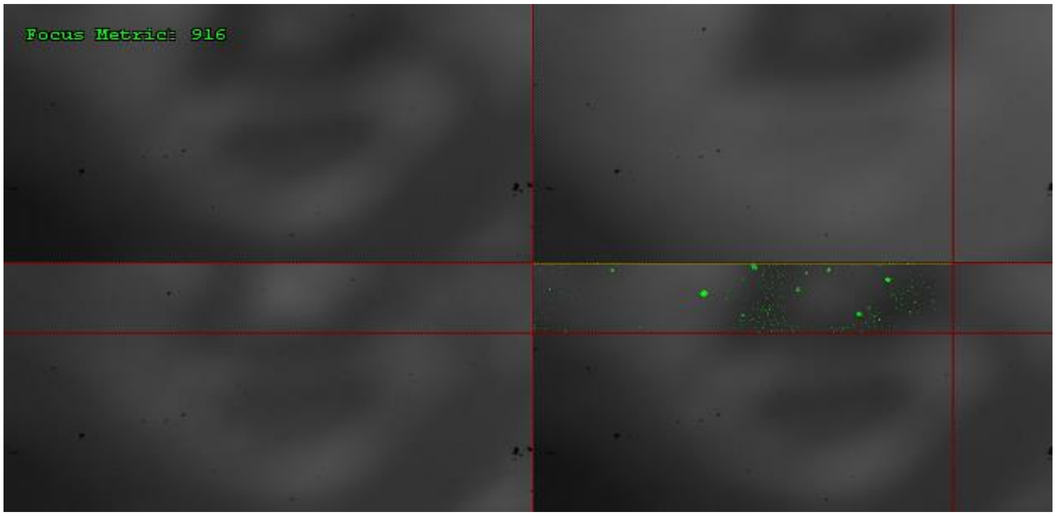

Figure 47: Image Made with Lensless Camera–Tiled Grayscale, Hardware Unswizzling

Finally, in Figure 48 we see the final data interpretation and setting adjustments.

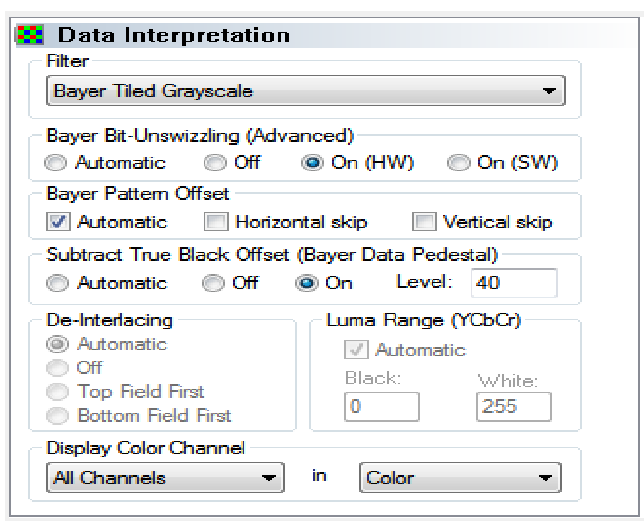

Figure 48: Data Interpretation

TESTING AND COMPARING RESULTS OF ORIGINAL IMAGE CAMERA WITH LENS WITH LENSLESS CAMERA IMAGE

In this section we move our face mask further from the camera.

Figure 49 shows the image of the mask–the data registers can be seen on the right of the original image made with a lensed camera.

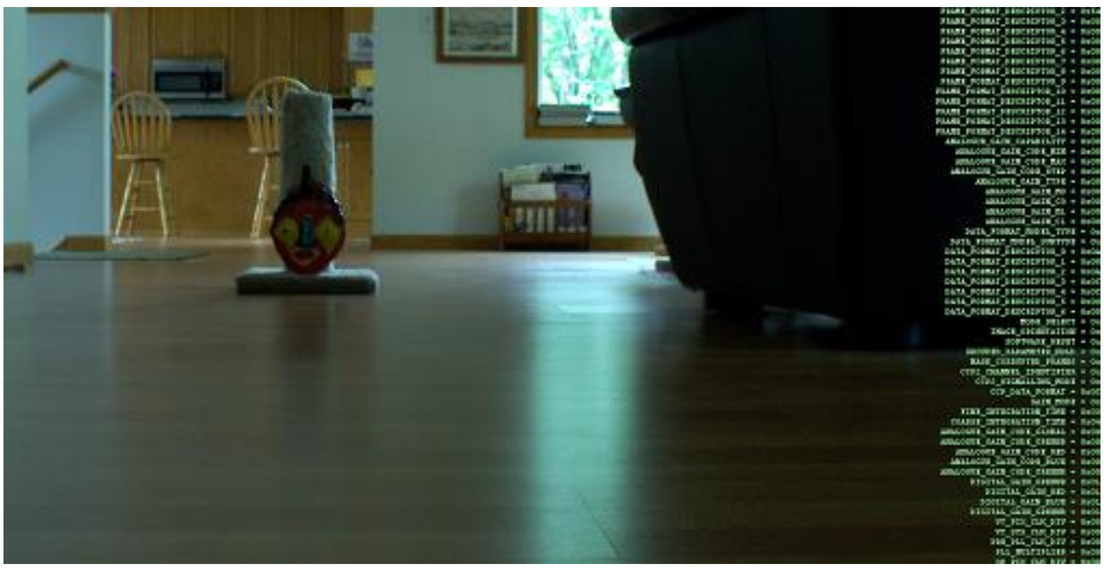

Figure 49: Original Image Lensed camera–Distance

In Figure 50, we see the edges of a chair in the background, magnified in the image. We clearly see edges, and we can manipulate the image. The lensless camera registers all motions and movements in front of it.

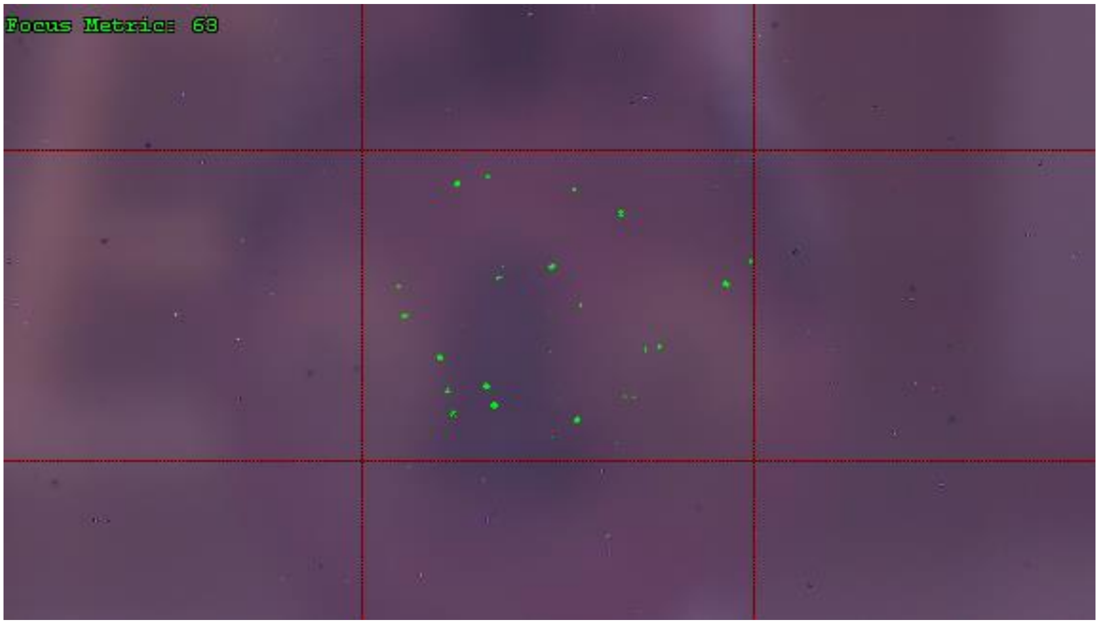

Figure 50: Lensless Camera Image–Distance, Magnified, Focus

Figure 51 shows our focused image obtained by the lensless camera, and color correction.

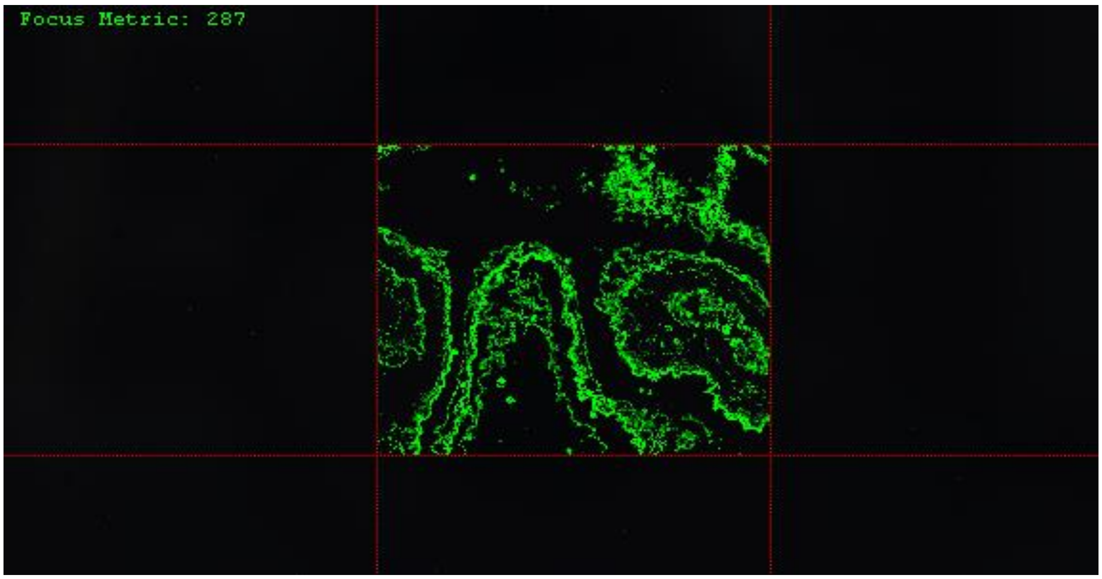

Figure 51: Original Image Made by Lensless Camera–Distance Focus, Color Correction

As we tried to improve the edges on our image, shown in Figure 51, we applied the following software correction setting parameters, shown in Figure 52: gamma set at 0.27, contrast at 2.00, blank correction at 2, saturation correction at 1.0, and aperture correction at 16.

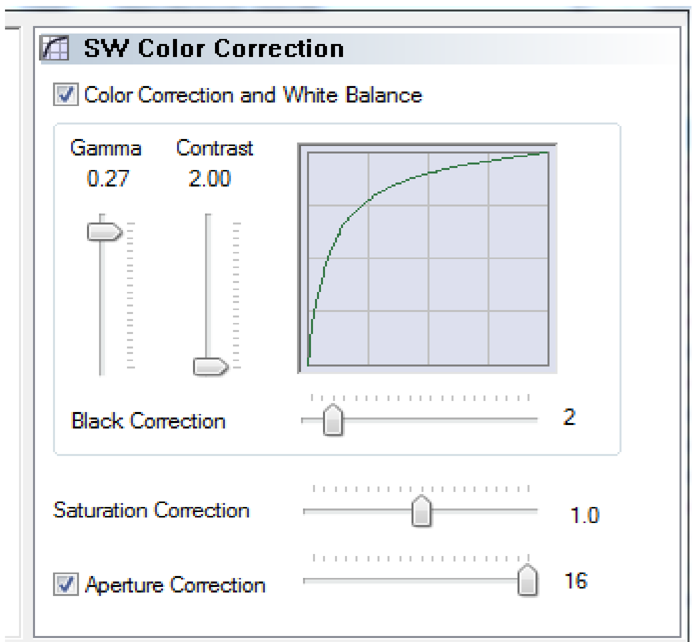

Figure 52: Color Correction Parameters

In Figure 53 we include live data from registers to register changes in movements.

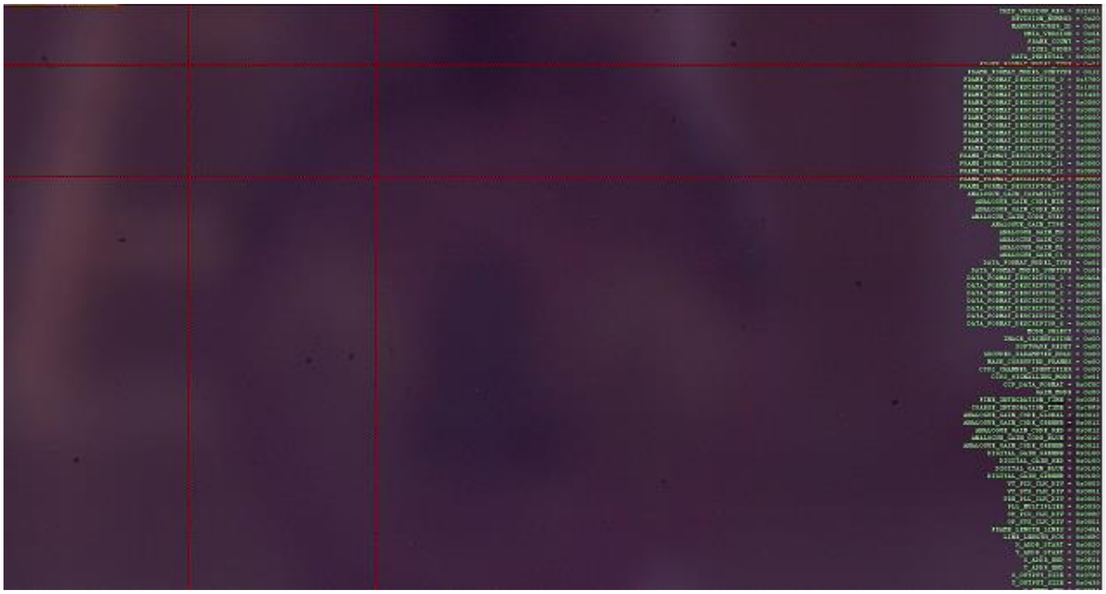

Figure 53: Lensless Camera Image–Distance with Live Data from Register

For this distance experiment, we present a tiled gray scale from the lensless camera image and the data life broadcast.

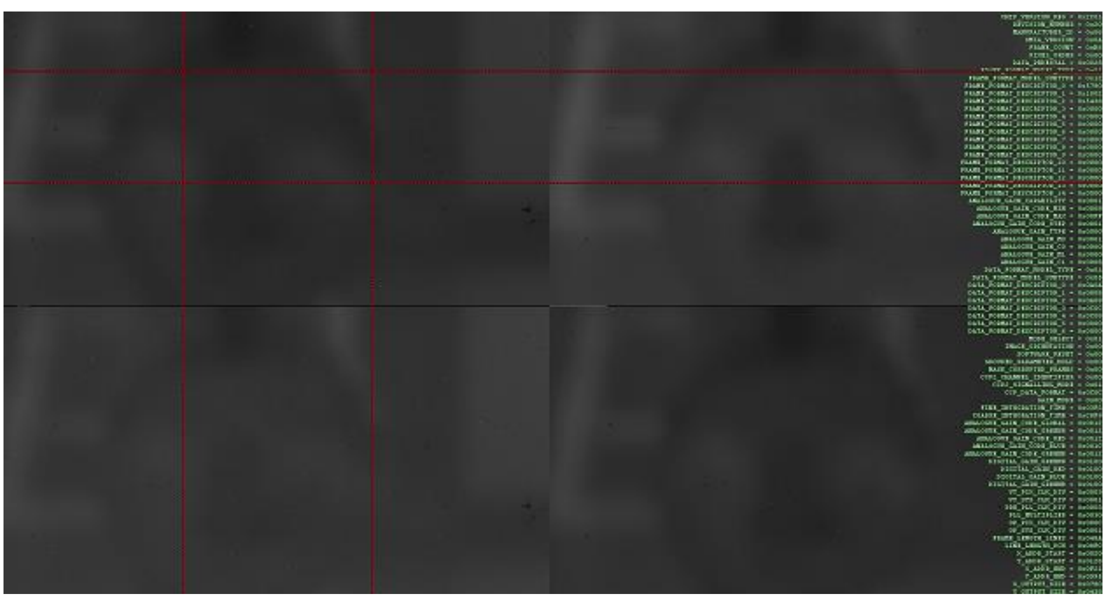

Figure 54: Lensless Camera Image–Distance Tiled Grayscale with live Datrom Register

In Figure 55, we see a black and white distance lensless camera image. We can see the chair and face edges.

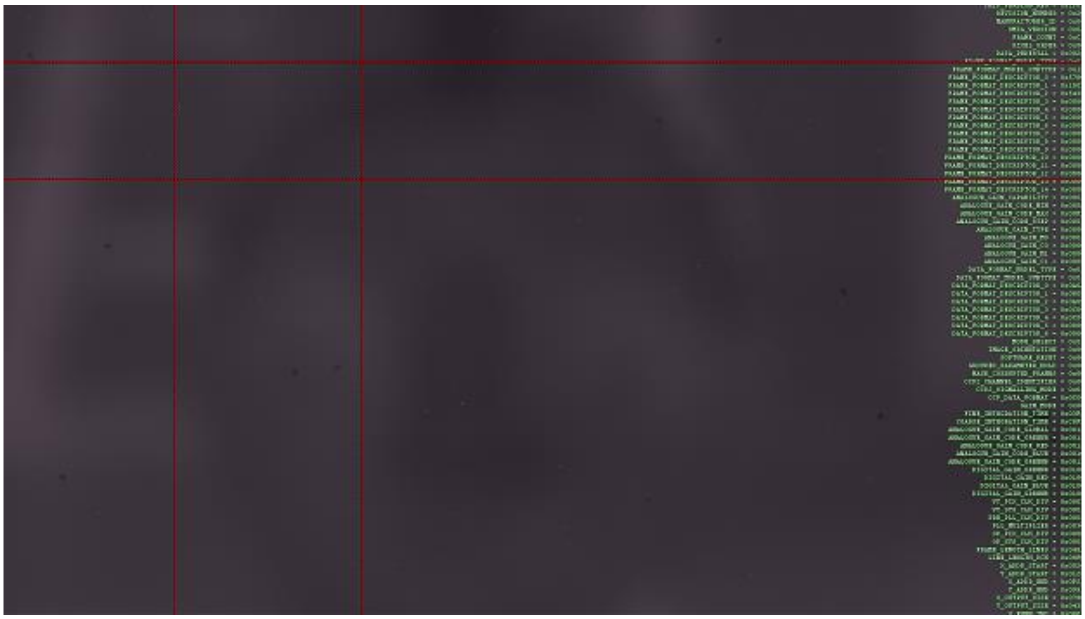

Figure 55: Lensless Camera Image–Distance, Grayscale with Data life from the Register

Figure 56 shows grayscale image parameters given by our implementation.

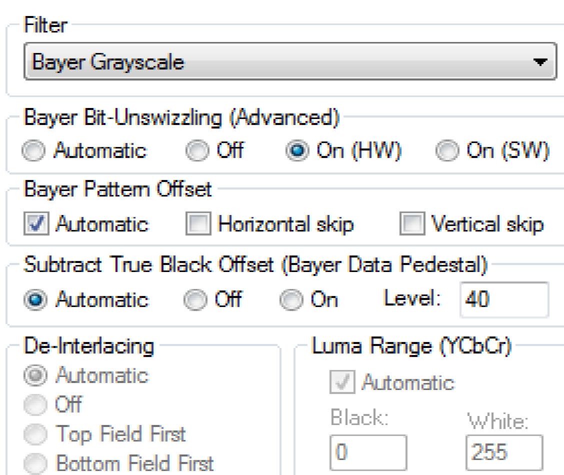

Figure 56: Grayscale Image Parameters

Chapter IV

LENSLESS CAMERA APPLICATION

In Chapter III, we conducted research, design, and experiments for a lensless camera based on an Aptina optical sensor. In Chapter IV, we describe our lensless camera application in robotics, and implementation in to the ROS.

The sensor inside the lensless camera is a highly integrated CMOS array with built-in edge enhancement and extraction. The built-in image processing tools enable a microcontroller to perform object detection and tracking. Assumptions for a robotic image processing system for detecting obstacles in its environment made, vary for different designs.

First, the lensless camera obtains the image, then the processing system progress the image. Once the edges are extracted from an image, the distance between the bottom edge and the bottom of the image can be measured.

We may mount the lensless camera on the front of the robot, a few inches or feet above the floor. The lensless camera is directed forward and down, and the floor does not have visible edges (i.e., the color is constant and there are no high-contrast seams), then the only edges should be the obstacles on the floor or a floor-to-wall transition in front of the robot. When the robot moves near a wall, and there is enough of a contrast between the wall and floor, an edge will be detected at that location in the image.

OBSTACLE AVOIDANCE

The robot can tell how far away the edge of the obstacle is by height in the image. If the image is divided into thirds (i.e., left third, center third, and right third), then the lowest edge in each third of the image gives the robot the distance it can move in that direction. Then, the robot can turn and move toward the farthest direction to avoid the closer obstacles. This "greatest third" approach is well suited for path following, since the greatest third of the image is probably the direction of the path. The lensless camera takes care of extracting the edges from the image, but additional processing is done by the microcontroller. For example, if we want to know an object's distance (or depth) from the robot, then we may need an algorithm to post-process the image and reduce the information down to a depth table. The index in a depth table represents a given x location. The entry at each index within the table could be written with the row of the lowest edge pixel in a given column. With few variations, this algorithm is implemented in most microcontrollers.

Let us briefly look into *the ordered list algorithm* of the depth-finding codes. We will start at the bottom-right corner of the image obtained by the lensless camera, and we count the number of pixels vertically until one is reached that surpasses a predefined threshold value. Put the row number (i.e., depth) of that pixel in a table, move to the next column to the left, and repeat the process. When the depths of all of the columns of the image have been recorded, send that information out of the universal asynchronous receiver / transmitter (UART).

OBJECT TRACKING

For object tracking movements, the lensless cameras find an object in the image. As previously described we assume that the lowest edge in the image, above some brightness threshold, is an object to be tracked. The microcontroller operates motors to pan and tilt the camera or robot itself to move the object to the center of the image. Advanced control of the motors prevents a slow response and overshooting. To execute object tracking, the microprocessor searches the image in RAM from the bottom up. When the microcontroller finds the first edge brighter than an assumed threshold value, it marks the x and y locations and measures the horizontal and vertical distance of this edge from the center of the image. Then, the microcontroller corrects and redirects the camera until the edge (object) is centered in the view.

Both the pan-tilt lensless camera head and the still-mounted lensless camera apparatus is suitable for tracking objects. Depending on lens sensitivity and robot response time we may track the objects moving at different speeds and different distances from the lensless camera. In our experiment we are tracking an object that is moving no faster than about 1 foot per second at a distance of 4 feet from the lensless camera. A helpful addition to the system would be a laser pointer; a bright point can be moved from one location to another almost instantaneously.

FOLLOWING THE OBJECT

Positioning the camera will enable us to control and move a robot. Our lensless camera setup is stationary with respect to the robot base. Our commands control the robot base, moving the robot in a desirable direction at a specified speed. These settings permit the mobile robot to find high-contrast objects and approach them. The lensless camera and the robot are able to search for objects by spiraling out in an ever-widening arc until an object is within view. When an object is detected, the robot faces the object and speeds towards it. The robot slows down gradually until it stops with the object located at the center of the camera image. As long as the object doesn't move too fast, the robot will continue to rotate, move forward, or move backward, to keep the object in the center of the image.

Once again, enhancing the output of the tracking system with a low-power laser pointer would benefit the robot movement. With a laser pointer addition we would be able to see a pulsating red dot, signaling where the robot is actually focusing. With proper settings and adjustments of the lensless camera and, the laser, a flat beam could be generated, visible as a horizontal line on any objects in its path. This line would be visible only at points in the field of view where an object is obstructing the beam. It would allow the lensless camera to have a greater ability to detect low-contrast objects against the floor background. Additionally, this approach would also help in finding low-contrast walls. As our experiment was done with a parallel high-speed PC connection we are able to explore pattern recognition in real-time, comparing the edge-extracted image to an edge database of known objects.

The use of two stationary lensless cameras as a pair could send both edge-extracted images via a high-speed parallel connection to a PC, allowing the PC to compare the two images and find matching patterns of edges. Pan angles of the two cameras would be fixed, and the matching pattern locations would allow the PC to determine the distance to certain objects based on the distance between the two cameras and the difference in pan angles.

Chapter V

CONCLUSION

During this research, we designed and conducted an experiment in a visualization lab based on an Aptina optical sensor, DevWare software, a Core i7 Intel processor, Windows 7 pro, and connecting hardware. For the image we used a colorful mask, shown in Figure 59.

Experiments were conducted using two lensless cameras with different diameter holes. Images were taken at two different distances. This resulted in four separate experiments: a lensless camera with a smaller apperture closer to the mask, the same camera farther from the mask, a lensless camera with a larger apperture closer to the mask, and the same camera farther from the mask.

Figure 57 illustrates the setups with the lensless camera closer to the mask. This was used for both apperture sizes.

Figure 57: Lensless Enclosure and Mask in Visualization Lab

In Figures 58 and 59 we may see the experiment setup in which the mask at a 7 foot distance from the camera.

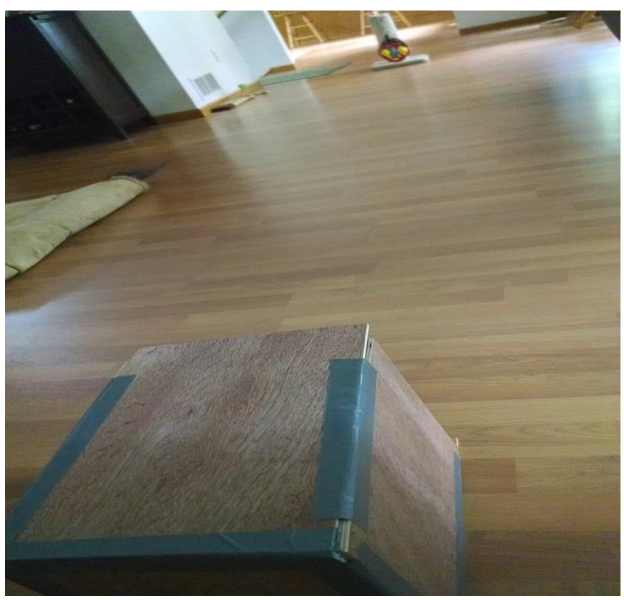

Figure 58: Distance Setting for Lensless Camera

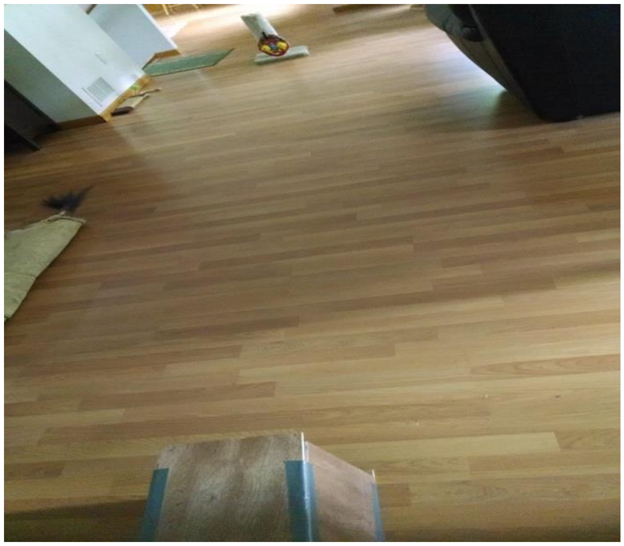

Figure 59: Distance Setting for Lensless Visualization Experiment

Figure 60 presents our image obtained by the lensed camera.

Figure 60: Original Image Mask–Lensed Camera

In Figure 61 we see our lensed mask image in focus metrics.

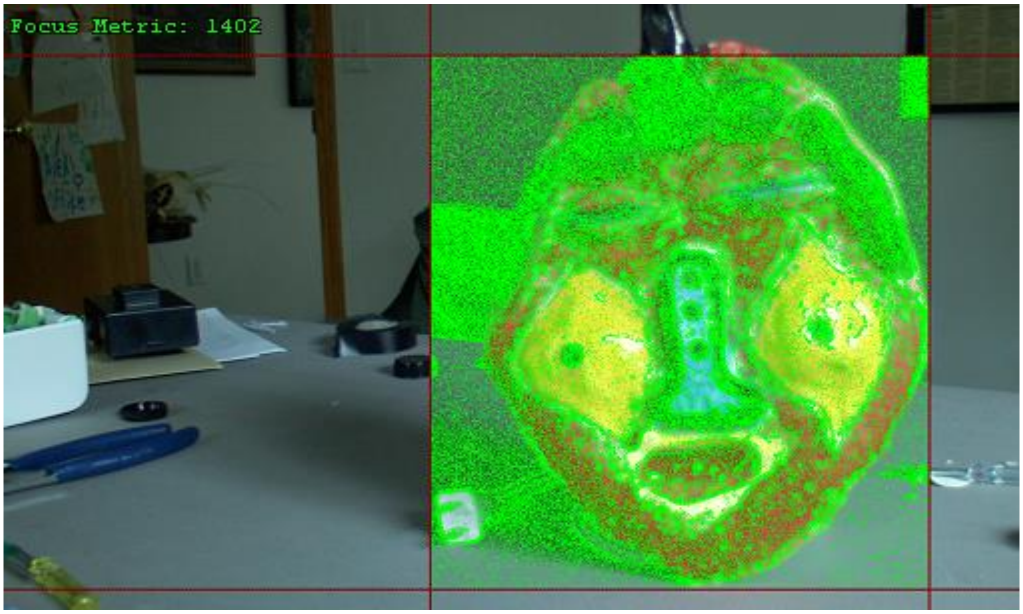

Figure 61: Original Image Mask–Focus Metric

Figure 62 shows image obtained by lensless camera with bigger apperture.

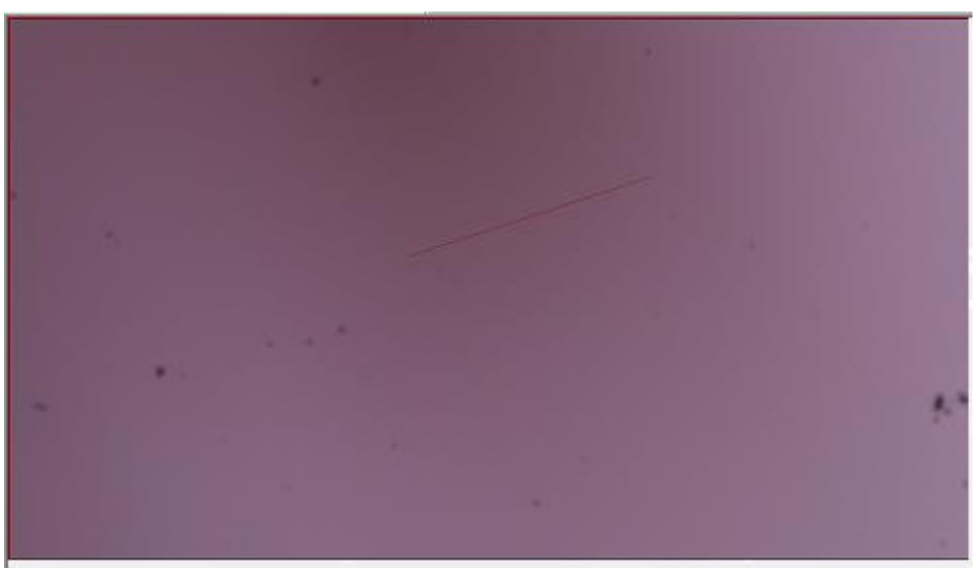

Figure 62: Image with Lensless Camera (using bigger apperture)

During our research we reduced the apperture and received an image of the mask shown in Figure 63. Here we can distinguish image shapes and colors.

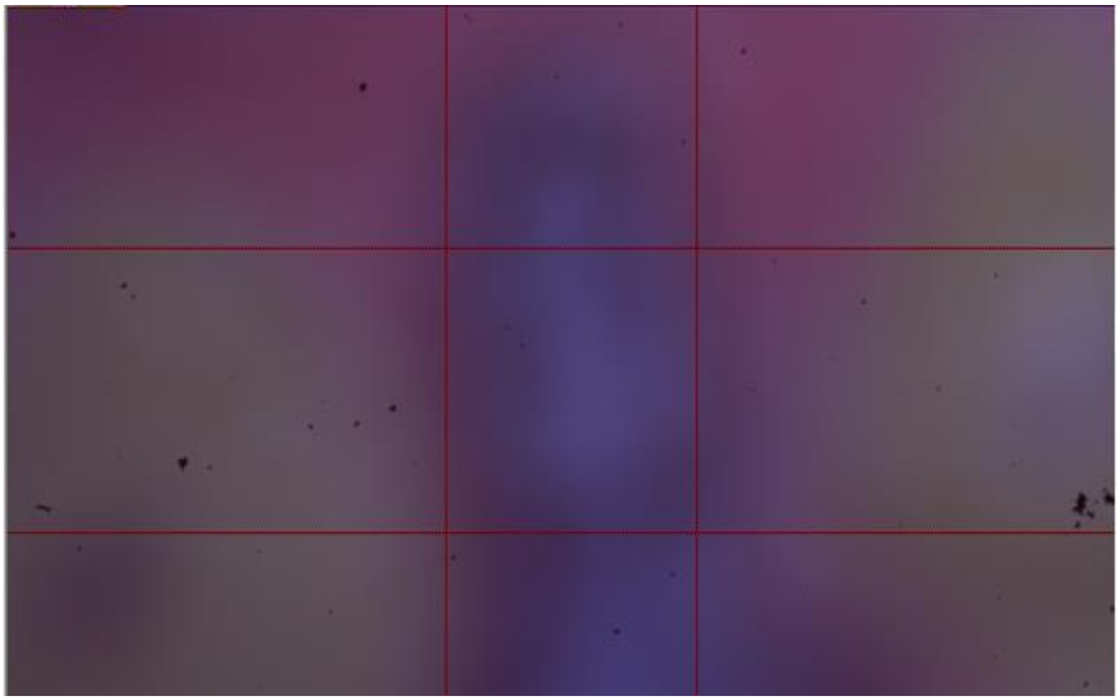

Figure 63: Image of the Mask Made with Lensless Camera Using Small Apperture

Figure 64 presents a more defined contour of the image.

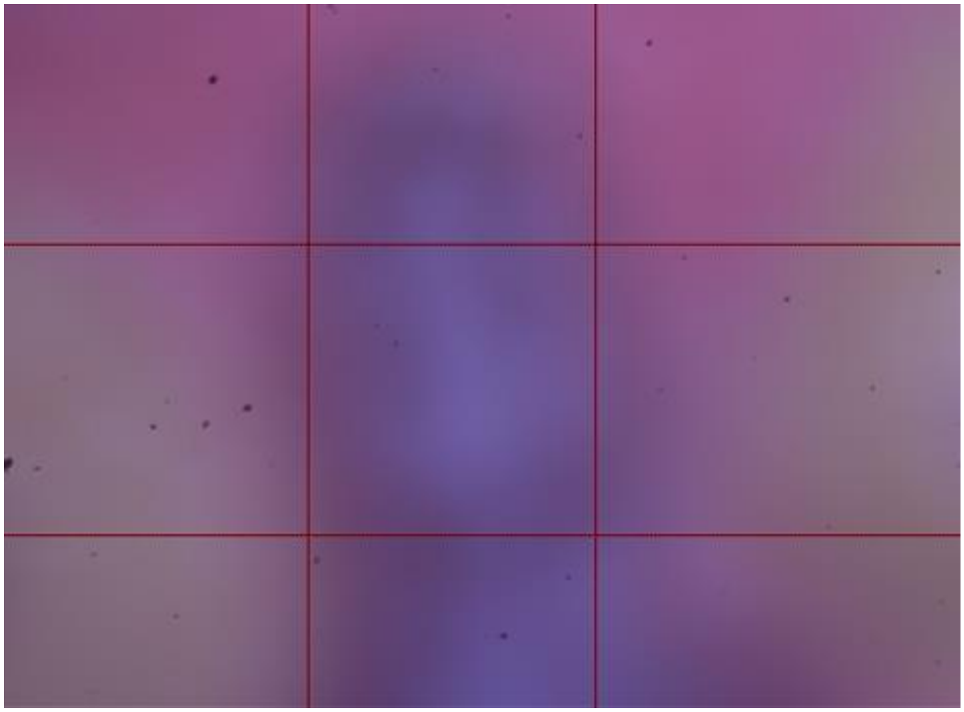

Figure 64: Image of the Mask Made with Lensless Camera

After applying manual gain, the magnified image of the face became more visible. See Figure 65.

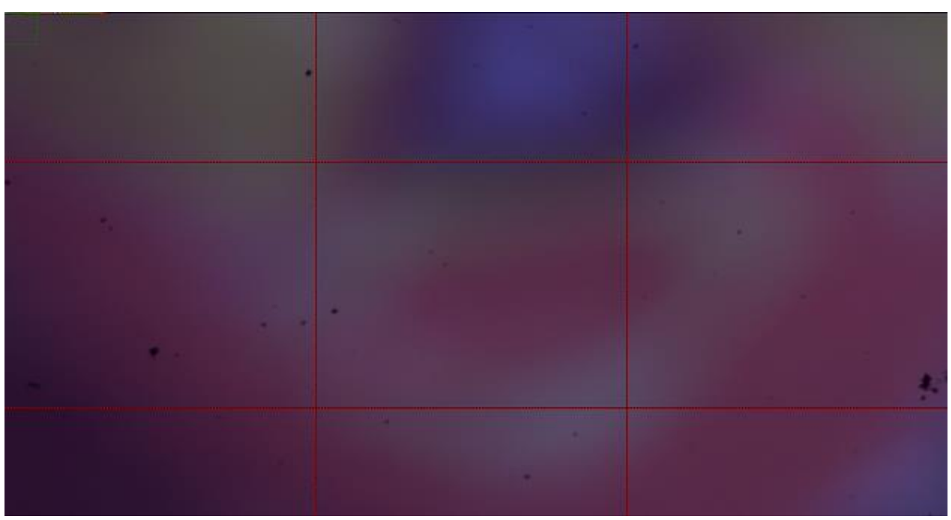

Figure 65: Image the Mask Made with Lensless Camera–Applied Manual Gain and High Exposure

The same image shown in figure 64 obtained with lensless camera become more evident as shapes and face contour in black and white become more evident, Figure 66.

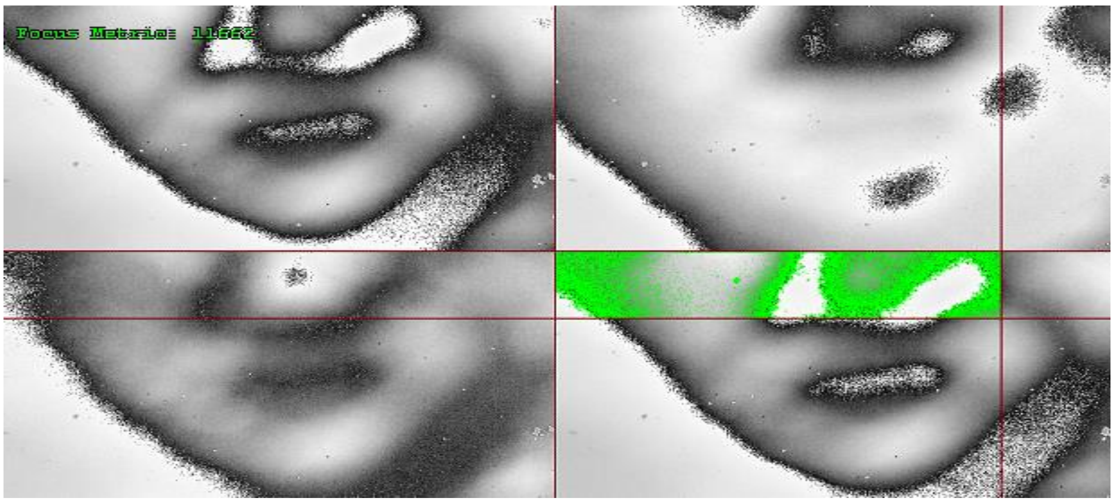

Figure 66: Image of the Mask Made with Lensless Camera–Focused, Tiled Grayscale

For comparison in Figure 67 we show the same image obtained by the lensed camera.

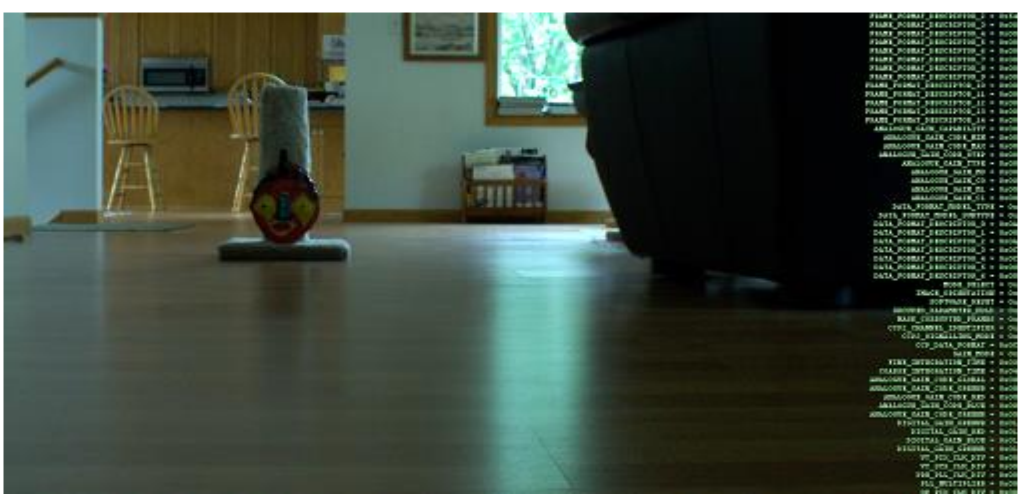

Figure 67: Original Image of the Mask Using Lensed Camera–Distance

In Figure 68 we show the color image obtained by the lensless camera with same data settings as the tiled grayscale image of the Mask in Figure 66.

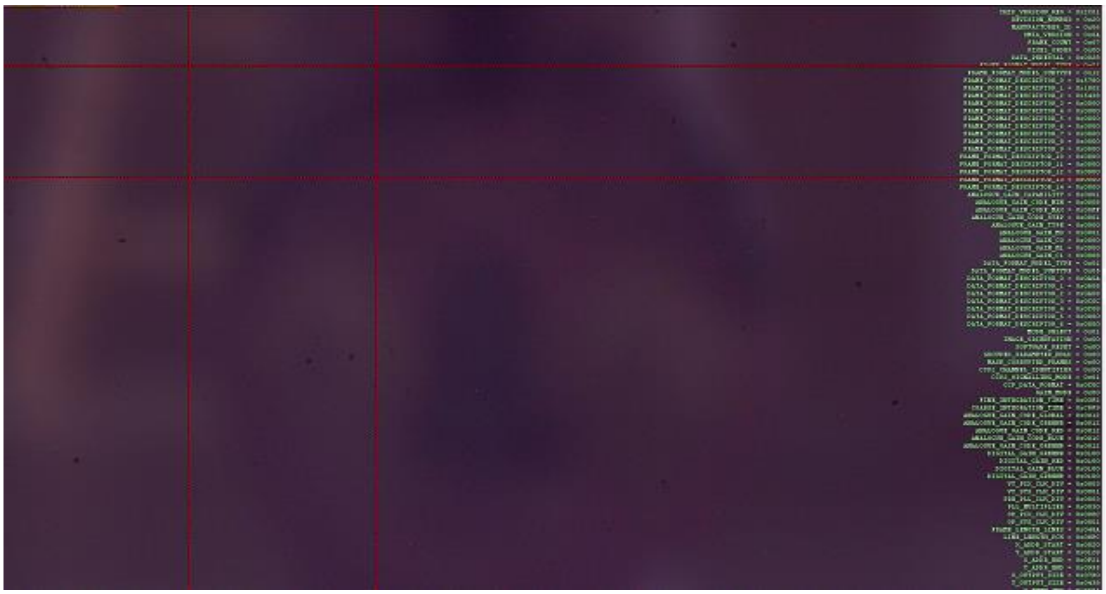

Figure 68: Lensless Camera Image–Distance with Life Data from Register

As result of our research, we may conclude that all movements, shapes, and obstacles may be seen by the image sensor of our lensless camera with adequate apparatus and it setting. Carefully adjusted parameters such as the camera apperture and the distance of the opening from the image sensor are important for obtaining a quality image.

REFERENCES

REFERENCES

[1] C. Goerzen, Z. Kong, and B. Mettler, “A survey of motion planning algorithms from the perspective of autonomous UAV guidance,” *Journal of Intelligent and Robotic Systems*, vol. 57, no. 1-4, 2010, pp. 65-100. 2010.

[2] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng,, http://www.robotics.stanford.edu/~ang/papers/icraoss09-ROS.pdf. April 2010

[3] H. Durrant-Whyte and T. Bailey, “Simultaneous localization and mapping: Part I,” *IEEE Robotics and Automation Magazine*, vol. 13, no. 2, pp. 99-110, 2006.

[4] J. Giesbrecht, “The global path planning for unmanned ground vehicles,” *The Technical Report*, Defense RnD Canada: Suffield, 2004.

[5] K. Dalamagkidis, K. P. Valavanis, and L. A. Piegl, “Autonomous autorotation of unmanned rotorcraft using nonlinear model”, 2010.

[6] P. Tompkins, A. Stentz, and D. Wettergreen, “Global path planning for Mars rover exploration,” In Proceedings of the IEEE Aerospace Conference– Predictive Control, *Journal of Intelligent and Robotic Systems,* vol. 57, no. 1-4, 2004, pp. 351-369.

[8] R. E. Kalman, “Contributions to the theory of optimal control,” *Bol. Soc. Mat. Mexicana*, pp. 102-119, 1960.

[9] O. Cappe, E. Moulines, and T. Ryden, *Inference in Hidden Markov Models*. Springer, 2005.

[10] H. Asada, Ford Professor of Engineering, MIT and M. Boulet, A Member of the Technical Staff in the Control Systems Engineering Group, MIT Lincoln Laboratory, “MIT–Robotic Summer Course 2013, http://courses.csail.mit.edu/6.141/

[11] http://www.ros.org/wiki/turtlebot_android/turtlebot_ android_follower.

[12] "Unmanned aircraft systems: Measuring progress and addressing potential privacy concerns would facilitate integration into the national airspace system," *Government Accountability Office Report GAO-12-981*, September 2012, http://www.fas.org/irp/gao/gao-12-981.pdf.

[13] A. E. A. Basu, "Global path planning using artificial potential fields," In *IEEE International Conference on Robotics and Automation*, 1989, pp. 316-321.

[14] R. Siegwart, R. I. Nourbakhsh, and D. Scaramuzza, Introduction To Autonomous Mobile Robots, Second edition. The MIT Press, 2011.

[15] http://news.cnet.com/2100-1001-984051.html[1], February 2003.

[16] http://www.youtube.com/watch?v=O0HLw3IkAOo, 2011.

[17] G. Voronoi, "Nouvelles applications des paramètres continus à la théorie des formes quadratiques," *Journal für die Reine und Angewandte Mathematik*, 1908.

[18] D. Chugo and S. Yokota, eds., *Introduction to Modern Robotics II*. Concept Press, Ltd, 2012,

[19] Katy S. Azoury, M. K. Warmuth "Relative Loss Bounds for On-line Density Estimation with the Exponential Family of Distributions", San Francisco, CA 94132 U.S.A, 2013.

[20] Jean-Paul Laumond, Nicolas Masard, Jean-Bernard Lasserre "Optimality in Robot Motion: Optimal versus Optimized Motion, ACM, Vol.57, NO 9 Septeember 2014.

---

[1] More also affirmed he never said transistor count would double every 18 months, as is commonly said. Initially, he said transistors on chip would double every year. He then recalibrated it to every 2 years in 1975. David House, an Intel executive at the time, noted that the changes would cause computer performance to double every 18 months.

APPENDICES

APPENDIX A

Data Image Normal Captured with Camera with Lens

Red MASK Image Normal captured with camera with lens

```
// Captured 12:43:23 - Wednesday, June 11, 2014
//
// Sensor info:
//      Width          = 960
//      Height         = 692
//      Image Format   = Bayer 12
//      Subformat      = GRBG
//      Sensor output  = 12 bits per pixel
//
// .RAW file info:
//      stored as      = 16 bits unsigned short
//      valid data     = xxxxBA98 76543210
//      Endianess      = little
//
// .BMP file info (24-bit windows bitmap file):
//      Width          = 960
//      Height         = 692
//
// Application version info:
//      Application Name = C:\Aptina Imaging\DevWare.exe
//      Application Version = 4.5.23.39222
//      Application Type = 4.5.23_Release
//      Application Date = 05/05/2014
```

```
//
// Camera info:
//        Product ID = 0x100D Version = 0xC1
//        Product Name = Aptina Imaging DEMO2X
//        Firmware Version = D.2A
//        Transport Name = USB 2.0
//        Chip 0 Name = DEMO2X C1
//                CHIP_VERSION_REG = 0x00C1
//        Chip 1 Name = High Speed Serial Adapter
//                VERSION = 0x0131
//                CHIP_VERSION_REG = 0x00CD
//                DATESTAMPREG = 0x304A
//                TIMESTAMPREG = 0x1016
//        Sensor Name = A-10030
//        Sensor Part Number = MT9J003
//        Sensor Version = REV3
//        Sensor Filename = C:\Aptina Imaging\sensor_data\MT9J003-REV3.xsdat
//
// Sensor Fuse info:
//        FuseID: D652768C5758C679
//        Revision: 2
//        Silicon Option: --
//        Customer Revision: 0x32
//
```

```
//  Windows OS info:
//          Display resolution = 1600x900 at 32bpp
//          OS Versioninfo = (6, 1, 7600, 2, , 0, 0)
//          Microsoft Windows Windows 7
//
//
//  Processor info:
//          2095 MHz
//          GenuineIntel
//          Intel(R) Core(TM) i7-3612QM CPU @ 2.10GHz
//          46 percent of memory is in use.
//          Memory 8063 MB (total)
//          Memory 4319 MB (available)
//
//  Display Devices:
//          Device 0: Intel(R) HD Graphics 4000
//
//  USB Host Controllers:
//          Service: usbehci
//                  Driver File: C:\Windows\system32\DRIVERS\usbehci.sys
//                  File Version: 6.1.7601.18328
//                  Device Desc 0:
PCI\VEN_8086&DEV_1E2D&SUBSYS_05641028&REV_04\3&11583659&0&D0
//                  Device Desc 1:
PCI\VEN_8086&DEV_1E26&SUBSYS_05641028&REV_04\3&11583659&0&E8
```

//

// Camera Driver Info:

// C:\Windows\System32\Drivers\USB64W7.SYS (3.4.1.20) - VendorID: 0x20FB (Aptina Imaging)

[ColorPipe State]

STATE= Display Zoom Percent, 33

STATE= Master Clock, 71000000

STATE= Update Sensor FPS, 1

STATE= Allow Update Sensor FPS, 1

STATE= Filter, 0

STATE= X Offset, 0

STATE= Y Offset, 0

STATE= Auto Offset, 1

STATE= CFA Pattern, 1

STATE= Gb-B Swap, 0

STATE= RGBC BiWindow, 2

STATE= Monochrome, 0

STATE= Bayer Quadrant, 0

STATE= Byte Swap, 0

STATE= RedBlue Swap, 0

STATE= True Black Scale, 4096

STATE= True Black Level, 40

STATE= True Black Enable, 2

STATE= Auto Luma Range, 1

STATE= Luma Lo, 0

STATE= Luma Hi, 255

STATE= Unswizzle Mode, 3

STATE= Swap 12-bit LSBs, 0

STATE= Column Repeat, 0

STATE= Orientation, 0

STATE= Deinterlace Mode, 3

STATE= Descramble Mode, 1

STATE= Special Pixel Mode, 0

STATE= Active Area Crop, 0

STATE= Output Channel, 0

STATE= Output BwColor, 0

STATE= X Bin, 1

STATE= Y Bin, 1

STATE= DVS Split Screen, 0

STATE= Stereo Merge, 0

STATE= Stereo Shift X, 0

STATE= Stereo Shift Y, 0

STATE= Stereo Colwise, 0

STATE= sRGB Color Standard, 0

STATE= Color Correction, 1

STATE= Gamma Correction, 45

STATE= Black Correct, 5

STATE= Saturation, 10

STATE= Contrast, 25

STATE= Aperture Enable, 1

STATE= Aperture, 4

STATE= Black CCM Kill Enable, 0

STATE= Black CCM Kill A, 240

STATE= Black CCM Kill B, 160

STATE= Black CCM Kill C, 80

STATE= Green Balance Enable, 0

STATE= Green Balance Apos, 128

STATE= Green Balance Bpos, 10

STATE= Green Balance Aneg, -128

STATE= Green Balance Bneg, 10

STATE= Auto Exposure, 1

STATE= Auto Exposure Target, 50

STATE= Auto Exposure Stability, 6

STATE= Auto Exposure Speed, 30

STATE= Auto Exposure Minimum FPS, 30

STATE= Auto Exposure Flicker Filter, 0

STATE= Auto Exposure Soft Limit, 33

STATE= Auto Exposure Soft Gain Limit, 40

STATE= Auto Exposure Gain Limit, 317

STATE= Auto Exposure Minimum Global Gain, 10

STATE= Auto Exposure Global Gain Limit, 10

STATE= Auto Exposure Software Gain Limit, 10

STATE= Auto Exposure Freeze Gains, 1

STATE= Auto Exposure Fade Saturation, 1

STATE= Auto Exposure Fade Aperture, 1

STATE= Auto Exposure Fade Target, 1

STATE= Auto Exposure Inner Zone, 50

STATE= Auto Exposure Outer Zone, 50

STATE= HDR AE Mode, 0

STATE= Software Gain, 1000

STATE= Mechanical Shutter Same, 1

STATE= Mechanical Shutter Time, 33333

STATE= Mechanical Shutter Delay, 0

STATE= Trigger Width, 0

STATE= White Balance, 1

STATE= WB Speed, 30

STATE= WB Adjust Gains, 0

STATE= WB Manual Position, 4

STATE= WB Manual RedGreen, 91

STATE= WB Manual BlueGreen, 61

STATE= WB Interpolate Saturation, 1

STATE= WB Normalize Matrix, 1

STATE= AWB Weight Map Method, 2

STATE= AWB Weight Map X Scale, 128

STATE= AWB Weight Map Y Scale, 256

STATE= AWB Weight Map X Shift, 32

STATE= AWB Weight Map Y Shift, 8

STATE= AWB Weight Map X Center, 1014

STATE= AWB Weight Map Y Center, 1009

STATE= AWB Weight Map Angle Sin, 48

STATE= AWB Weight Map Angle Cos, 43

STATE= AWB Weight Map Luma Low, 4

STATE= AWB Weight Map Luma High, 251

STATE= Minimum Gain, 1000

STATE= Show Min Gain As 1, 0

STATE= Default Relative Red Gain, 1000

STATE= Default Relative Blue Gain, 1570

STATE= Relative Red Gain, 1069

STATE= Relative Blue Gain, 3050

STATE= Lens Correction Enable, 0

STATE= Lens Correction Falloff, 100

STATE= Lens Correction Falloff R, 100

STATE= Lens Correction Falloff G1, 100

STATE= Lens Correction Falloff G2, 100

STATE= Lens Correction Falloff B, 100

STATE= Lens Correction Overlay, 0

STATE= Lens Correction Center X, 1920

STATE= Lens Correction Center Y, 1374

STATE= Lens Correction Coeff Prec, 16

STATE= Lens Correction Lens Radius, 0

STATE= Lens Correction Luma Only, 0

STATE= Lens Center Red X, 1920

STATE= Lens Center Red Y, 1374

STATE= Lens Center Green1 X, 1920

STATE= Lens Center Green1 Y, 1374

STATE= Lens Center Green2 X, 1920

STATE= Lens Center Green2 Y, 1374

STATE= Lens Center Blue X, 1920

STATE= Lens Center Blue Y, 1374

STATE= Lens Center Overlay, 0

STATE= Lens Sim Sensor, 0

STATE= Lens Sim Sensor Rev, 0

STATE= Lens Sim Enable Pwq, 0

STATE= Lens Sim Enable Poly, 0

STATE= Noise Removal, 1

STATE= Noise Removal Level, 14

STATE= Noise Removal Depth, 2

STATE= Noise Removal K1, 2000

STATE= Noise Removal K2, 1800

STATE= Noise Removal K3, 1000

STATE= Noise Removal Edges, 1

STATE= Noise Removal Kernel, 0

STATE= Optical Black, 0

STATE= Optical Black Row Filter, 1

STATE= Optical Black 1 Row Start, 0

STATE= Optical Black 1 Row End, 0

STATE= Optical Black 2 Row Start, 0

STATE= Optical Black 2 Row End, 0

STATE= Optical Black 1 Column Start, 0

STATE= Optical Black 1 Column End, 0

STATE= Optical Black 2 Column Start, 0

STATE= Optical Black 2 Column End, 0

STATE= ALTM Enable, 0

STATE= Defect Enable, 1

STATE= Defect Max, 10000

STATE= Defect Auto Defect Correction, 0

STATE= Flash Lamp, 0

STATE= Still Global Reset, 0

STATE= Global Reset Bulb, 0

STATE= Continuous GRR, 0

STATE= Num Capture Frames, 1

STATE= Still Mode, 0

STATE= Still Hold, 1

STATE= Still Capture Average, 0

STATE= Still Capture Timeout, 5

STATE= Still HalfPress, 0

STATE= Delay before snap, 0

STATE= Save 24bpp BMP, 1

STATE= Save RAW, 1

STATE= Save TXT, 1

STATE= Save HEX, 0

STATE= Save ITX, 0

STATE= Save CCR, 0

STATE= Save DXR, 0

STATE= Save 48bpp COLOR TIFF, 0

STATE= Save JPEG, 0

STATE= Save RAW JPEG, 0

STATE= Save BMP Info, 0

STATE= JPEG Quality (1-100), 98

STATE= Save RAW PNG, 0

STATE= Save PNG, 0

STATE= Save DNG, 0

STATE= Save SS, 0

STATE= Save Selection Rectangle, 0

STATE= ICC Profile, 0

STATE= Video Screen Capture, 1

STATE= VidCap Play FPS, 15000

STATE= VidCap Auto Play FPS, 1

STATE= VidCap Format, 0

STATE= VidCap FileName Increment, 1

STATE= RAM Capture, 0

STATE= RAM Capture MB, 128

STATE= RAM Capture Cycle, 1

STATE= Preview Recording, 1

STATE= WB Xenon Red Gain, 1024

STATE= WB Xenon Blue Gain, 1024

STATE= WB Led Red Gain, 1024

STATE= WB Led Blue Gain, 1024

STATE= MAE Overlay, 0

STATE= Noise Image Type, 0

STATE= Noise Frames, 50

STATE= Noise Defects, 0

STATE= Strip FSP, 1

STATE= Check Thumbnail Table, 0

STATE= CRA Overlay, 0

STATE= Allow FAR Access, 1

STATE= Pure Raw, 0

STATE= Sensor Orientation, 0

STATE= Clarity6, 3

STATE= Clarity7, 1

STATE= Dynamic Range LSB, 0

STATE= Dynamic Range MSB, 31

STATE= AI Repack, 1

STATE= AR1820 REV12 GHR Workaround, 0

STATE= Deinterleave HDR, 3

STATE= HDR AE Histogram High Percent, 0.99

STATE= HDR AE Histogram High Target, 0.75

STATE= HDR AE Scene Brightness Offset, 8

STATE= HDR AE Max Percent Adjust, 0

STATE= HDR AE Luma Target, 40 80 200 600 1000 1400 1800 2200

STATE= AWB Incandescent, 1.886 -1.248 0.362 -0.128 1.329 -0.201 -0.082 -0.775 1.857

STATE= AWB Sun, 2.036 -1.055 0.019 -0.086 1.351 -0.265 0.007 -0.551 1.544

STATE= AWB Incandescent Gain, 0.896 1.987

STATE= AWB Sun Gain, 1.691 0.957

STATE= WB Custom, 1 0 0 0 1 0 0 0 1

STATE= WB Custom Xenon, 1 0 0 0 1 0 0 0 1

STATE= WB Custom Led, 1 0 0 0 1 0 0 0 1

STATE= AWB Weight Map, 0 3 8224 4096 1 273 4866 4368 34 8995 8739 12816 308 17204 12612 17168 546 21349 4915 13361 291 21333 8241 12832 3 8755 4610 4640 17 275 0 0

STATE= Optical Black Level Rect, 0 0 0 0

STATE= Active Area Rect, -1 -1 -1 -1

STATE= Lens Curve Red,
C04192013743F800000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000

STATE= Lens Curve Green1,
C04192013743F800000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000

STATE= Lens Curve Green2,
C04192013743F800000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000

STATE= Lens Curve Blue,
C04192013743F800000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000

STATE= RGBC Sigma_S, 1

STATE= RGBC Sigma_I, 0.005

STATE= RGBC Smooth_Th, 0.1

STATE= ALTM Peak Percent, 0.99

STATE= ALTM Sharp S, 1

STATE= HW Gain, 1.57813

STATE= HW Global Gain, 1

STATE= HW Red Gain, 1.67188

STATE= HW Green1 Gain, 1.57813

STATE= HW Green2 Gain, 1.57813

STATE= HW Blue Gain, 4.8125

STATE= HW Exposure Time, 0.0303211

STATE= VidCap File, C:\Users\Aleksandar\Documents\Aptina Imaging\video.avi

STATE= Clarity2, 1

STATE= Clarity3, 0.04

STATE= Clarity4, 0

STATE= Clarity5, 0.34

STATE= Chroma Filter Absolute Add Corners, 0

STATE= Clarity23, 1 7 0

STATE= Clarity24, 24aff 1249e

STATE= Clarity8, 0.028

STATE= Clarity9, 0.12

STATE= Clarity25, 2 2

STATE= Clarity11, 0

STATE= Clarity12, 0.117

STATE= Clarity10, 0.05

STATE= Clarity22, 0.025

STATE= Clarity13, 0.04

STATE= Clarity14, 0.2

STATE= Clarity15, 8

STATE= Black Balance, 0 0 0 0

STATE= Clarity18, 1

STATE= Clarity17, 2

STATE= Clarity20, 0.24 0.17

STATE= Clarity19, 0.1 0.1

STATE= Clarity21, 0.05

STATE= Clarity16, 0.01

STATE= Clarity26, 0.1

STATE= Clarity27, 1

STATE= Post Gain, 1

STATE= NIR Factor Red, 1

STATE= NIR Factor Green, 1

STATE= NIR Factor Blue, 1

STATE= NIR Coeff Red, 0

STATE= NIR Coeff Green, 0

STATE= NIR Coeff Blue, 0

STATE= NIR WB Custom, 1 0 0 0 1 0 0 0 1

STATE= NIR WB Gains, 1 1 1

STATE= NIR Global Gain, 1

[Raw Image Format]

IMAGE= 960, 692, BAYER-12

[Register State]

```
REG= 0x0000, 0x2C01    // CHIP_VERSION_REG
REG= 0x0002, 0x20      // REVISION_NUMBER
REG= 0x0003, 0x06      // MANUFACTURER_ID
REG= 0x0004, 0x0A      // SMIA_VERSION
REG= 0x0005, 0xFF      // FRAME_COUNT
REG= 0x0006, 0x00      // PIXEL_ORDER
REG= 0x0008, 0x0028    // DATA_PEDESTAL
REG= 0x0040, 0x01      // FRAME_FORMAT_MODEL_TYPE
REG= 0x0041, 0x12      // FRAME_FORMAT_MODEL_SUBTYPE
REG= 0x0042, 0x53C0    // FRAME_FORMAT_DESCRIPTOR_0
REG= 0x0044, 0x1002    // FRAME_FORMAT_DESCRIPTOR_1
REG= 0x0046, 0x52B2    // FRAME_FORMAT_DESCRIPTOR_2
REG= 0x0048, 0x0000    // FRAME_FORMAT_DESCRIPTOR_3
REG= 0x004A, 0x0000    // FRAME_FORMAT_DESCRIPTOR_4
REG= 0x004C, 0x0000    // FRAME_FORMAT_DESCRIPTOR_5
REG= 0x004E, 0x0000    // FRAME_FORMAT_DESCRIPTOR_6
REG= 0x0050, 0x0000    // FRAME_FORMAT_DESCRIPTOR_7
REG= 0x0052, 0x0000    // FRAME_FORMAT_DESCRIPTOR_8
```

```
REG= 0x0054, 0x0000   // FRAME_FORMAT_DESCRIPTOR_9
REG= 0x0056, 0x0000   // FRAME_FORMAT_DESCRIPTOR_10
REG= 0x0058, 0x0000   // FRAME_FORMAT_DESCRIPTOR_11
REG= 0x005A, 0x0000   // FRAME_FORMAT_DESCRIPTOR_12
REG= 0x005C, 0x0000   // FRAME_FORMAT_DESCRIPTOR_13
REG= 0x005E, 0x0000   // FRAME_FORMAT_DESCRIPTOR_14
REG= 0x0080, 0x0001   // ANALOGUE_GAIN_CAPABILITY
REG= 0x0084, 0x0008   // ANALOGUE_GAIN_CODE_MIN
REG= 0x0086, 0x00FF   // ANALOGUE_GAIN_CODE_MAX
REG= 0x0088, 0x0001   // ANALOGUE_GAIN_CODE_STEP
REG= 0x008A, 0x0000   // ANALOGUE_GAIN_TYPE
REG= 0x008C, 0x0001   // ANALOGUE_GAIN_M0
REG= 0x008E, 0x0000   // ANALOGUE_GAIN_C0
REG= 0x0090, 0x0000   // ANALOGUE_GAIN_M1
REG= 0x0092, 0x0008   // ANALOGUE_GAIN_C1
REG= 0x00C0, 0x01     // DATA_FORMAT_MODEL_TYPE
REG= 0x00C1, 0x05     // DATA_FORMAT_MODEL_SUBTYPE
REG= 0x00C2, 0x0A0A   // DATA_FORMAT_DESCRIPTOR_0
REG= 0x00C4, 0x0808   // DATA_FORMAT_DESCRIPTOR_1
REG= 0x00C6, 0x0A08   // DATA_FORMAT_DESCRIPTOR_2
REG= 0x00C8, 0x0C0C   // DATA_FORMAT_DESCRIPTOR_3
REG= 0x00CA, 0x0C08   // DATA_FORMAT_DESCRIPTOR_4
REG= 0x00CC, 0x0000   // DATA_FORMAT_DESCRIPTOR_5
REG= 0x00CE, 0x0000   // DATA_FORMAT_DESCRIPTOR_6
```

```
REG= 0x0100, 0x01     // MODE_SELECT
REG= 0x0101, 0x00     // IMAGE_ORIENTATION
REG= 0x0103, 0x00     // SOFTWARE_RESET
REG= 0x0104, 0x00     // GROUPED_PARAMETER_HOLD
REG= 0x0105, 0x00     // MASK_CORRUPTED_FRAMES
REG= 0x0110, 0x00     // CCP2_CHANNEL_IDENTIFIER
REG= 0x0111, 0x01     // CCP2_SIGNALLING_MODE
REG= 0x0112, 0x0C0C   // CCP_DATA_FORMAT
REG= 0x0120, 0x00     // GAIN_MODE
REG= 0x0200, 0x03F2   // FINE_INTEGRATION_TIME
REG= 0x0202, 0x0114   // COARSE_INTEGRATION_TIME
REG= 0x0204, 0x000C   // ANALOGUE_GAIN_CODE_GLOBAL
REG= 0x0206, 0x000C   // ANALOGUE_GAIN_CODE_GREENR
REG= 0x0208, 0x000D   // ANALOGUE_GAIN_CODE_RED
REG= 0x020A, 0x0026   // ANALOGUE_GAIN_CODE_BLUE
REG= 0x020C, 0x000C   // ANALOGUE_GAIN_CODE_GREENB
REG= 0x020E, 0x0100   // DIGITAL_GAIN_GREENR
REG= 0x0210, 0x0100   // DIGITAL_GAIN_RED
REG= 0x0212, 0x0100   // DIGITAL_GAIN_BLUE
REG= 0x0214, 0x0100   // DIGITAL_GAIN_GREENB
REG= 0x0300, 0x0003   // VT_PIX_CLK_DIV
REG= 0x0302, 0x0001   // VT_SYS_CLK_DIV
REG= 0x0304, 0x0003   // PRE_PLL_CLK_DIV
REG= 0x0306, 0x0030   // PLL_MULTIPLIER
```

```
REG= 0x0308, 0x000C    // OP_PIX_CLK_DIV
REG= 0x030A, 0x0001    // OP_SYS_CLK_DIV
REG= 0x0340, 0x04C9    // FRAME_LENGTH_LINES
REG= 0x0342, 0x1E78    // LINE_LENGTH_PCK
REG= 0x0344, 0x0018    // X_ADDR_START
REG= 0x0346, 0x0008    // Y_ADDR_START
REG= 0x0348, 0x0F15    // X_ADDR_END
REG= 0x034A, 0x0AC1    // Y_ADDR_END
REG= 0x034C, 0x03C0    // X_OUTPUT_SIZE
REG= 0x034E, 0x02B2    // Y_OUTPUT_SIZE
REG= 0x0380, 0x0001    // X_EVEN_INC
REG= 0x0382, 0x0003    // X_ODD_INC
REG= 0x0384, 0x0001    // Y_EVEN_INC
REG= 0x0386, 0x0003    // Y_ODD_INC
REG= 0x0400, 0x0002    // SCALING_MODE
REG= 0x0402, 0x0000    // SPATIAL_SAMPLING
REG= 0x0404, 0x001F    // SCALE_M
REG= 0x0406, 0x0010    // SCALE_N
REG= 0x0500, 0x0001    // COMPRESSION_MODE
REG= 0x0600, 0x0000    // TEST_PATTERN_MODE
REG= 0x0602, 0x0000    // TEST_DATA_RED
REG= 0x0604, 0x0000    // TEST_DATA_GREENR
REG= 0x0606, 0x0000    // TEST_DATA_BLUE
REG= 0x0608, 0x0000    // TEST_DATA_GREENB
```

```
REG= 0x060A, 0x0000    // HORIZONTAL_CURSOR_WIDTH
REG= 0x060C, 0x0000    // HORIZONTAL_CURSOR_POSITION
REG= 0x060E, 0x0000    // VERTICAL_CURSOR_WIDTH
REG= 0x0610, 0x0000    // VERTICAL_CURSOR_POSITION
REG= 0x1000, 0x0001    // INTEGRATION_TIME_CAPABILITY
REG= 0x1004, 0x0000    // COARSE_INTEGRATION_TIME_MIN
REG= 0x1006, 0x0001    // COARSE_INTEGRATION_TIME_MAX_MARGIN
REG= 0x1008, 0x03F2    // FINE_INTEGRATION_TIME_MIN
REG= 0x100A, 0x027E    // FINE_INTEGRATION_TIME_MAX_MARGIN
REG= 0x1080, 0x0001    // DIGITAL_GAIN_CAPABILITY
REG= 0x1084, 0x0100    // DIGITAL_GAIN_MIN
REG= 0x1086, 0x0700    // DIGITAL_GAIN_MAX
REG= 0x1088, 0x0100    // DIGITAL_GAIN_STEP_SIZE
REG= 0x1100, 0x40000000        // MIN_EXT_CLK_FREQ_MHZ
REG= 0x1104, 0x42800000        // MAX_EXT_CLK_FREQ_MHZ
REG= 0x1108, 0x0001    // MIN_PRE_PLL_CLK_DIV
REG= 0x110A, 0x0040    // MAX_PRE_PLL_CLK_DIV
REG= 0x110C, 0x40800000        // MIN_PLL_IP_FREQ_MHZ
REG= 0x1110, 0x41C00000        // MAX_PLL_IP_FREQ_MHZ
REG= 0x1114, 0x0020    // MIN_PLL_MULTIPLIER
REG= 0x1116, 0x0180    // MAX_PLL_MULTIPLIER
REG= 0x1118, 0x43C00000        // MIN_PLL_OP_FREQ_MHZ
REG= 0x111C, 0x44400000        // MAX_PLL_OP_FREQ_MHZ
REG= 0x1120, 0x0001    // MIN_VT_SYS_CLK_DIV
```

REG= 0x1122, 0x0001 // MAX_VT_SYS_CLK_DIV

REG= 0x1124, 0x41C00000 // MIN_VT_SYS_CLK_FREQ_MHZ

REG= 0x1128, 0x44400000 // MAX_VT_SYS_CLK_FREQ_MHZ

REG= 0x112C, 0x4019999A // MIN_VT_PIX_CLK_FREQ_MHZ

REG= 0x1130, 0x42C00000 // MAX_VT_PIX_CLK_FREQ_MHZ

REG= 0x1134, 0x0004 // MIN_VT_PIX_CLK_DIV

REG= 0x1136, 0x0006 // MAX_VT_PIX_CLK_DIV

REG= 0x1140, 0x0091 // MIN_FRAME_LENGTH_LINES

REG= 0x1142, 0xFFFF // MAX_FRAME_LENGTH_LINES

REG= 0x1144, 0x0670 // MIN_LINE_LENGTH_PCK

REG= 0x1146, 0xFFFE // MAX_LINE_LENGTH_PCK

REG= 0x1148, 0x046E // MIN_LINE_BLANKING_PCK

REG= 0x114A, 0x008F // MIN_FRAME_BLANKING_LINES

REG= 0x1160, 0x0001 // MIN_OP_SYS_CLK_DIV

REG= 0x1162, 0x0001 // MAX_OP_SYS_CLK_DIV

REG= 0x1164, 0x4199999A // MIN_OP_SYS_CLK_FREQ_MHZ

REG= 0x1168, 0x44400000 // MAX_OP_SYS_CLK_FREQ_MHZ

REG= 0x116C, 0x0008 // MIN_OP_PIX_CLK_DIV

REG= 0x116E, 0x000C // MAX_OP_PIX_CLK_DIV

REG= 0x1170, 0x4019999A // MIN_OP_PIX_CLK_FREQ_MHZ

REG= 0x1174, 0x42C00000 // MAX_OP_PIX_CLK_FREQ_MHZ

REG= 0x1180, 0x0018 // X_ADDR_MIN

REG= 0x1182, 0x0000 // Y_ADDR_MIN

REG= 0x1184, 0x0F27 // X_ADDR_MAX

```
REG= 0x1186, 0x0ACB    // Y_ADDR_MAX
REG= 0x11C0, 0x0001    // MIN_EVEN_INC
REG= 0x11C2, 0x0001    // MAX_EVEN_INC
REG= 0x11C4, 0x0001    // MIN_ODD_INC
REG= 0x11C6, 0x0003    // MAX_ODD_INC
REG= 0x1200, 0x0002    // SCALING_CAPABILITY
REG= 0x1204, 0x0010    // SCALER_M_MIN
REG= 0x1206, 0x0080    // SCALER_M_MAX
REG= 0x1208, 0x0010    // SCALER_N_MIN
REG= 0x120A, 0x0010    // SCALER_N_MAX
REG= 0x1300, 0x0001    // COMPRESSION_CAPABILITY
REG= 0x1400, 0x0242    // MATRIX_ELEMENT_REDINRED
REG= 0x1402, 0xFF00    // MATRIX_ELEMENT_GREENINRED
REG= 0x1404, 0xFFBE    // MATRIX_ELEMENT_BLUEINRED
REG= 0x1406, 0xFFB4    // MATRIX_ELEMENT_REDINGREEN
REG= 0x1408, 0x0200    // MATRIX_ELEMENT_GREENINGREEN
REG= 0x140A, 0xFF4D    // MATRIX_ELEMENT_BLUEINGREEN
REG= 0x140C, 0xFFF1    // MATRIX_ELEMENT_REDINBLUE
REG= 0x140E, 0xFF34    // MATRIX_ELEMENT_GREENINBLUE
REG= 0x1410, 0x01DC    // MATRIX_ELEMENT_BLUEINBLUE
REG= 0x3000, 0x2C01    // MODEL_ID_
REG= 0x3002, 0x0008    // Y_ADDR_START_
REG= 0x3004, 0x0018    // X_ADDR_START_
REG= 0x3006, 0x0AC1    // Y_ADDR_END_
```

```
REG= 0x3008, 0x0F15     // X_ADDR_END_
REG= 0x300A, 0x04C9     // FRAME_LENGTH_LINES_
REG= 0x300C, 0x1E78     // LINE_LENGTH_PCK_
REG= 0x3010, 0x009C     // FINE_CORRECTION
REG= 0x3012, 0x0114     // COARSE_INTEGRATION_TIME_
REG= 0x3014, 0x03F2     // FINE_INTEGRATION_TIME_
REG= 0x3016, 0x0121     // ROW_SPEED
REG= 0x3018, 0x0000     // EXTRA_DELAY
REG= 0x301A, 0x001C     // RESET_REGISTER
REG= 0x301C, 0x01       // MODE_SELECT_
REG= 0x301D, 0x00       // IMAGE_ORIENTATION_
REG= 0x301E, 0x0028     // DATA_PEDESTAL_
REG= 0x3021, 0x00       // SOFTWARE_RESET_
REG= 0x3022, 0x00       // GROUPED_PARAMETER_HOLD_
REG= 0x3023, 0x00       // MASK_CORRUPTED_FRAMES_
REG= 0x3024, 0x00       // PIXEL_ORDER_
REG= 0x3026, 0xFFFF     // GPI_STATUS
REG= 0x3028, 0x000C     // ANALOGUE_GAIN_CODE_GLOBAL_
REG= 0x302A, 0x000C     // ANALOGUE_GAIN_CODE_GREENR_
REG= 0x302C, 0x000D     // ANALOGUE_GAIN_CODE_RED_
REG= 0x302E, 0x0026     // ANALOGUE_GAIN_CODE_BLUE_
REG= 0x3030, 0x000C     // ANALOGUE_GAIN_CODE_GREENB_
REG= 0x3032, 0x0100     // DIGITAL_GAIN_GREENR_
REG= 0x3034, 0x0100     // DIGITAL_GAIN_RED_
```

```
REG= 0x3036, 0x0100    // DIGITAL_GAIN_BLUE_
REG= 0x3038, 0x0100    // DIGITAL_GAIN_GREENB_
REG= 0x303A, 0x0A      // SMIA_VERSION_
REG= 0x303B, 0xFF      // FRAME_COUNT_
REG= 0x303C, 0x0000    // FRAME_STATUS
REG= 0x3040, 0x38C3    // READ_MODE
REG= 0x3044, 0x0590    // RESERVED_MFR_3044
REG= 0x3046, 0x0600    // FLASH
REG= 0x3048, 0x0008    // FLASH_COUNT
REG= 0x304A, 0x0020    // RESERVED_MFR_304A
REG= 0x304C, 0x0200    // RESERVED_MFR_304C
REG= 0x304E, 0x0000    // RESERVED_MFR_304E
REG= 0x3050, 0x0000    // RESERVED_MFR_3050
REG= 0x3052, 0x2174    // RESERVED_MFR_3052
REG= 0x3054, 0x0000    // RESERVED_MFR_3054
REG= 0x3056, 0x1065    // GREEN1_GAIN
REG= 0x3058, 0x1C4D    // BLUE_GAIN
REG= 0x305A, 0x106B    // RED_GAIN
REG= 0x305C, 0x1065    // GREEN2_GAIN
REG= 0x305E, 0x1065    // GLOBAL_GAIN
REG= 0x3060, 0x1500    // RESERVED_MFR_3060
REG= 0x3062, 0x0000    // RESERVED_MFR_3062
REG= 0x3064, 0x0905    // RESERVED_MFR_3064
REG= 0x3066, 0x0000    // RESERVED_MFR_3066
```

```
REG= 0x3068, 0x0000    // RESERVED_MFR_3068
REG= 0x306A, 0x0000    // DATAPATH_STATUS
REG= 0x306C, 0x0842    // RESERVED_MFR_306C
REG= 0x306E, 0x90B0    // DATAPATH_SELECT
REG= 0x3070, 0x0000    // TEST_PATTERN_MODE_
REG= 0x3072, 0x0000    // TEST_DATA_RED_
REG= 0x3074, 0x0000    // TEST_DATA_GREENR_
REG= 0x3076, 0x0000    // TEST_DATA_BLUE_
REG= 0x3078, 0x0000    // TEST_DATA_GREENB_
REG= 0x307A, 0x0000    // TEST_RAW_MODE
REG= 0x3080, 0x0000    // RESERVED_MFR_3080
REG= 0x30A0, 0x0001    // X_EVEN_INC_
REG= 0x30A2, 0x0003    // X_ODD_INC_
REG= 0x30A4, 0x0001    // Y_EVEN_INC_
REG= 0x30A6, 0x0003    // Y_ODD_INC_
REG= 0x30A8, 0x0017    // CALIB_GREEN1_ASC1
REG= 0x30AA, 0x0042    // CALIB_BLUE_ASC1
REG= 0x30AC, 0x0005    // CALIB_RED_ASC1
REG= 0x30AE, 0x0006    // CALIB_GREEN2_ASC1
REG= 0x30B0, 0x0003    // RESERVED_MFR_30B0
REG= 0x30B2, 0x8000    // RESERVED_MFR_30B2
REG= 0x30B4, 0x01FF    // RESERVED_MFR_30B4
REG= 0x30BC, 0x1000    // CALIB_GLOBAL
REG= 0x30C0, 0x0120    // CALIB_CONTROL
```

```
REG= 0x30C2, 0x0017    // CALIB_GREEN1
REG= 0x30C4, 0x0042    // CALIB_BLUE
REG= 0x30C6, 0x0005    // CALIB_RED
REG= 0x30C8, 0x0005    // CALIB_GREEN2
REG= 0x30CA, 0x8004    // RESERVED_MFR_30CA
REG= 0x30CC, 0x0000    // RESERVED_MFR_30CC
REG= 0x30CE, 0x0FFF    // RESERVED_MFR_30CE
REG= 0x30D0, 0x0FFE    // RESERVED_MFR_30D0
REG= 0x30D2, 0x0FFE    // RESERVED_MFR_30D2
REG= 0x30D4, 0x9080    // RESERVED_MFR_30D4
REG= 0x30D6, 0x0800    // RESERVED_MFR_30D6
REG= 0x30D8, 0x0000    // RESERVED_MFR_30D8
REG= 0x30DA, 0x0000    // RESERVED_MFR_30DA
REG= 0x30DC, 0x0000    // RESERVED_MFR_30DC
REG= 0x3130, 0x0F1F    // RESERVED_MFR_3130
REG= 0x3132, 0x0F1F    // RESERVED_MFR_3132
REG= 0x3134, 0x9F13    // RESERVED_MFR_3134
REG= 0x3136, 0x0404    // RESERVED_MFR_3136
REG= 0x3138, 0x4409    // RESERVED_MFR_3138
REG= 0x313A, 0x0000    // RESERVED_MFR_313A
REG= 0x313C, 0x0000    // RESERVED_MFR_313C
REG= 0x313E, 0x0000    // RESERVED_MFR_313E
REG= 0x315C, 0x0000    // RESERVED_MFR_315C
REG= 0x315E, 0x0000    // GLOBAL_SEQ_TRIGGER
```

```
REG= 0x3160, 0x0098    // GLOBAL_RST_END
REG= 0x3162, 0x00A8    // GLOBAL_SHUTTER_START
REG= 0x3164, 0x0000    // GLOBAL_SHUTTER_START2
REG= 0x3166, 0x00B8    // GLOBAL_READ_START
REG= 0x3168, 0x0000    // GLOBAL_READ_START2
REG= 0x316A, 0x0000    // RESERVED_MFR_316A
REG= 0x316C, 0x0429    // RESERVED_MFR_316C
REG= 0x316E, 0x0400    // RESERVED_MFR_316E
REG= 0x3170, 0x00E5    // RESERVED_MFR_3170
REG= 0x3172, 0x0501    // RESERVED_MFR_3172
REG= 0x3174, 0x8000    // RESERVED_MFR_3174
REG= 0x3176, 0x0000    // RESERVED_MFR_3176
REG= 0x3178, 0x0070    // RESERVED_MFR_3178
REG= 0x318A, 0x0008    // RESERVED_MFR_318A
REG= 0x318C, 0x0FFD    // RESERVED_MFR_318C
REG= 0x318E, 0x0FFE    // RESERVED_MFR_318E
REG= 0x3190, 0x0009    // RESERVED_MFR_3190
REG= 0x31A0, 0x0101    // DESCRIPTOR_0
REG= 0x31A2, 0x0201    // DESCRIPTOR_1
REG= 0x31A4, 0x0202    // DESCRIPTOR_2
REG= 0x31A6, 0x0301    // DESCRIPTOR_3
REG= 0x31A8, 0x0302    // DESCRIPTOR_4
REG= 0x31AA, 0x0304    // DESCRIPTOR_5
REG= 0x31AC, 0x0000    // DESCRIPTOR_6
```

```
REG= 0x31AE, 0x0304    // SERIAL_FORMAT
REG= 0x31B0, 0x0063    // FRAME_PREAMBLE
REG= 0x31B2, 0x0039    // LINE_PREAMBLE
REG= 0x31B4, 0x0D57    // MIPI_TIMING_0
REG= 0x31B6, 0x0B10    // MIPI_TIMING_1
REG= 0x31B8, 0x010D    // MIPI_TIMING_2
REG= 0x31BA, 0x050D    // MIPI_TIMING_3
REG= 0x31BC, 0x000B    // MIPI_TIMING_4
REG= 0x31BE, 0xC003    // RESERVED_MFR_31BE
REG= 0x31C0, 0x0000    // HISPI_TIMING
REG= 0x31C2, 0xFFFF    // RESERVED_MFR_31C2
REG= 0x31C4, 0xF555    // RESERVED_MFR_31C4
REG= 0x31C6, 0x0000    // HISPI_CONTROL_STATUS
REG= 0x31C8, 0x0000    // RESERVED_MFR_31C8
REG= 0x31CA, 0x0000    // RESERVED_MFR_31CA
REG= 0x31CC, 0x0000    // RESERVED_MFR_31CC
REG= 0x31CE, 0x0000    // RESERVED_MFR_31CE
REG= 0x31DA, 0x0000    // RESERVED_MFR_31DA
REG= 0x31DC, 0x0000    // RESERVED_MFR_31DC
REG= 0x31DE, 0x0000    // RESERVED_MFR_31DE
REG= 0x31E0, 0x0003    // RESERVED_MFR_31E0
REG= 0x31E2, 0x0000    // RESERVED_MFR_31E2
REG= 0x31E4, 0x0000    // RESERVED_MFR_31E4
REG= 0x31E8, 0x0000    // HORIZONTAL_CURSOR_POSITION_
```

```
REG= 0x31EA, 0x0000    // VERTICAL_CURSOR_POSITION_
REG= 0x31EC, 0x0000    // HORIZONTAL_CURSOR_WIDTH_
REG= 0x31EE, 0x0000    // VERTICAL_CURSOR_WIDTH_
REG= 0x31F2, 0x0000    // I2C_IDS_MIPI_DEFAULT
REG= 0x31F4, 0x0000    // RESERVED_MFR_31F4
REG= 0x31F6, 0x0000    // RESERVED_MFR_31F6
REG= 0x31F8, 0x0000    // RESERVED_MFR_31F8
REG= 0x31FA, 0x0000    // RESERVED_MFR_31FA
REG= 0x31FC, 0x3020    // I2C_IDS
REG= 0x31FE, 0x0032    // RESERVED_MFR_31FE
REG= 0x3600, 0x0850    // P_GR_P0Q0
REG= 0x3602, 0x0850    // P_GR_P0Q1
REG= 0x3604, 0x0850    // P_GR_P0Q2
REG= 0x3606, 0x0850    // P_GR_P0Q3
REG= 0x3608, 0x0850    // P_GR_P0Q4
REG= 0x360A, 0x0850    // P_RD_P0Q0
REG= 0x360C, 0x0850    // P_RD_P0Q1
REG= 0x360E, 0x0850    // P_RD_P0Q2
REG= 0x3610, 0x0850    // P_RD_P0Q3
REG= 0x3612, 0x0850    // P_RD_P0Q4
REG= 0x3614, 0x0850    // P_BL_P0Q0
REG= 0x3616, 0x0850    // P_BL_P0Q1
REG= 0x3618, 0x0850    // P_BL_P0Q2
REG= 0x361A, 0x0850    // P_BL_P0Q3
```

```
REG= 0x361C, 0x0850    // P_BL_P0Q4
REG= 0x361E, 0x0850    // P_GB_P0Q0
REG= 0x3620, 0x0850    // P_GB_P0Q1
REG= 0x3622, 0x0850    // P_GB_P0Q2
REG= 0x3624, 0x0850    // P_GB_P0Q3
REG= 0x3626, 0x0850    // P_GB_P0Q4
REG= 0x3640, 0xC20B    // P_GR_P1Q0
REG= 0x3642, 0xC20B    // P_GR_P1Q1
REG= 0x3644, 0xC20B    // P_GR_P1Q2
REG= 0x3646, 0xC20B    // P_GR_P1Q3
REG= 0x3648, 0xC20B    // P_GR_P1Q4
REG= 0x364A, 0xC20B    // P_RD_P1Q0
REG= 0x364C, 0xC20B    // P_RD_P1Q1
REG= 0x364E, 0xC20B    // P_RD_P1Q2
REG= 0x3650, 0xC20B    // P_RD_P1Q3
REG= 0x3652, 0xC20B    // P_RD_P1Q4
REG= 0x3654, 0xC20B    // P_BL_P1Q0
REG= 0x3656, 0xC20B    // P_BL_P1Q1
REG= 0x3658, 0xC20B    // P_BL_P1Q2
REG= 0x365A, 0xC20B    // P_BL_P1Q3
REG= 0x365C, 0xC20B    // P_BL_P1Q4
REG= 0x365E, 0xC20B    // P_GB_P1Q0
REG= 0x3660, 0xC20B    // P_GB_P1Q1
REG= 0x3662, 0xC20B    // P_GB_P1Q2
```

```
REG= 0x3664, 0xC20B    // P_GB_P1Q3
REG= 0x3666, 0xC20B    // P_GB_P1Q4
REG= 0x3680, 0x6CAA    // P_GR_P2Q0
REG= 0x3682, 0x6CAA    // P_GR_P2Q1
REG= 0x3684, 0x6CAA    // P_GR_P2Q2
REG= 0x3686, 0x6CAA    // P_GR_P2Q3
REG= 0x3688, 0x6CAA    // P_GR_P2Q4
REG= 0x368A, 0x6CAA    // P_RD_P2Q0
REG= 0x368C, 0x6CAA    // P_RD_P2Q1
REG= 0x368E, 0x6CAA    // P_RD_P2Q2
REG= 0x3690, 0x6CAA    // P_RD_P2Q3
REG= 0x3692, 0x6CAA    // P_RD_P2Q4
REG= 0x3694, 0x6CAA    // P_BL_P2Q0
REG= 0x3696, 0x6CAA    // P_BL_P2Q1
REG= 0x3698, 0x6CAA    // P_BL_P2Q2
REG= 0x369A, 0x6CAA    // P_BL_P2Q3
REG= 0x369C, 0x6CAA    // P_BL_P2Q4
REG= 0x369E, 0x6CAA    // P_GB_P2Q0
REG= 0x36A0, 0x6CAA    // P_GB_P2Q1
REG= 0x36A2, 0x6CAA    // P_GB_P2Q2
REG= 0x36A4, 0x6CAA    // P_GB_P2Q3
REG= 0x36A6, 0x6CAA    // P_GB_P2Q4
REG= 0x36C0, 0x9A0A    // P_GR_P3Q0
REG= 0x36C2, 0x9A0A    // P_GR_P3Q1
```

```
REG= 0x36C4, 0x9A0A   // P_GR_P3Q2
REG= 0x36C6, 0x9A0A   // P_GR_P3Q3
REG= 0x36C8, 0x9A0A   // P_GR_P3Q4
REG= 0x36CA, 0x9A0A   // P_RD_P3Q0
REG= 0x36CC, 0x9A0A   // P_RD_P3Q1
REG= 0x36CE, 0x9A0A   // P_RD_P3Q2
REG= 0x36D0, 0x9A0A   // P_RD_P3Q3
REG= 0x36D2, 0x9A0A   // P_RD_P3Q4
REG= 0x36D4, 0x9A0A   // P_BL_P3Q0
REG= 0x36D6, 0x9A0A   // P_BL_P3Q1
REG= 0x36D8, 0x9A0A   // P_BL_P3Q2
REG= 0x36DA, 0x9A0A   // P_BL_P3Q3
REG= 0x36DC, 0x9A0A   // P_BL_P3Q4
REG= 0x36DE, 0x9A0A   // P_GB_P3Q0
REG= 0x36E0, 0x9A0A   // P_GB_P3Q1
REG= 0x36E2, 0x9A0A   // P_GB_P3Q2
REG= 0x36E4, 0x9A0A   // P_GB_P3Q3
REG= 0x36E6, 0x9A0A   // P_GB_P3Q4
REG= 0x3700, 0xEE6C   // P_GR_P4Q0
REG= 0x3702, 0xEE6C   // P_GR_P4Q1
REG= 0x3704, 0xEE6C   // P_GR_P4Q2
REG= 0x3706, 0xEE6C   // P_GR_P4Q3
REG= 0x3708, 0xEE6C   // P_GR_P4Q4
REG= 0x370A, 0xEE6C   // P_RD_P4Q0
```

```
REG= 0x370C, 0xEE6C    // P_RD_P4Q1
REG= 0x370E, 0xEE6C    // P_RD_P4Q2
REG= 0x3710, 0xEE6C    // P_RD_P4Q3
REG= 0x3712, 0xEE6C    // P_RD_P4Q4
REG= 0x3714, 0xEE6C    // P_BL_P4Q0
REG= 0x3716, 0xEE6C    // P_BL_P4Q1
REG= 0x3718, 0xEE6C    // P_BL_P4Q2
REG= 0x371A, 0xEE6C    // P_BL_P4Q3
REG= 0x371C, 0xEE6C    // P_BL_P4Q4
REG= 0x371E, 0xEE6C    // P_GB_P4Q0
REG= 0x3720, 0xEE6C    // P_GB_P4Q1
REG= 0x3722, 0xEE6C    // P_GB_P4Q2
REG= 0x3724, 0xEE6C    // P_GB_P4Q3
REG= 0x3726, 0xEE6C    // P_GB_P4Q4
REG= 0x3780, 0x8000    // POLY_SC_ENABLE
REG= 0x3782, 0x07BC    // POLY_ORIGIN_C
REG= 0x3784, 0x0544    // POLY_ORIGIN_R
REG= 0x3800, 0x0000    // RESERVED_MFR_3800
REG= 0x3802, 0x0000    // RESERVED_MFR_3802
REG= 0x3804, 0x0000    // RESERVED_MFR_3804
REG= 0x3806, 0x0000    // RESERVED_MFR_3806
REG= 0x3808, 0x0000    // RESERVED_MFR_3808
REG= 0x380A, 0x0000    // RESERVED_MFR_380A
REG= 0x380C, 0x0000    // RESERVED_MFR_380C
```

```
REG= 0x380E, 0x0000    // RESERVED_MFR_380E
REG= 0x3810, 0x0000    // RESERVED_MFR_3810
REG= 0x3812, 0x0000    // RESERVED_MFR_3812
REG= 0x3814, 0x0000    // RESERVED_MFR_3814
REG= 0x3816, 0x0000    // RESERVED_MFR_3816
REG= 0x3818, 0x0000    // RESERVED_MFR_3818
REG= 0x381A, 0x0000    // RESERVED_MFR_381A
REG= 0x381C, 0x0000    // RESERVED_MFR_381C
REG= 0x381E, 0x0000    // RESERVED_MFR_381E
REG= 0x3820, 0x0000    // RESERVED_MFR_3820
REG= 0x3822, 0x0000    // RESERVED_MFR_3822
REG= 0x3824, 0x0000    // RESERVED_MFR_3824
REG= 0x3826, 0x0000    // RESERVED_MFR_3826
REG= 0x3828, 0x0000    // RESERVED_MFR_3828
REG= 0x382A, 0x0000    // RESERVED_MFR_382A
REG= 0x382C, 0x0000    // RESERVED_MFR_382C
REG= 0x382E, 0x0000    // RESERVED_MFR_382E
REG= 0x3830, 0x0000    // RESERVED_MFR_3830
REG= 0x3832, 0x0000    // RESERVED_MFR_3832
REG= 0x3834, 0x0000    // RESERVED_MFR_3834
REG= 0x3836, 0x0000    // RESERVED_MFR_3836
REG= 0x3838, 0x0000    // RESERVED_MFR_3838
REG= 0x383A, 0x0000    // RESERVED_MFR_383A
REG= 0x383C, 0x0000    // RESERVED_MFR_383C
```

```
REG= 0x383E, 0x0000    // RESERVED_MFR_383E
REG= 0x3840, 0x0000    // RESERVED_MFR_3840
REG= 0x3842, 0x0000    // RESERVED_MFR_3842
REG= 0x3844, 0x0000    // RESERVED_MFR_3844
REG= 0x3846, 0x0000    // RESERVED_MFR_3846
REG= 0x3848, 0x0000    // RESERVED_MFR_3848
REG= 0x384A, 0x0000    // RESERVED_MFR_384A
REG= 0x384C, 0x0000    // RESERVED_MFR_384C
REG= 0x384E, 0x0000    // RESERVED_MFR_384E
REG= 0x3850, 0x0000    // RESERVED_MFR_3850
REG= 0x3852, 0x0000    // RESERVED_MFR_3852
REG= 0x3854, 0x0000    // RESERVED_MFR_3854
REG= 0x3856, 0x0000    // RESERVED_MFR_3856
REG= 0x3858, 0x0000    // RESERVED_MFR_3858
REG= 0x385A, 0x0000    // RESERVED_MFR_385A
REG= 0x385C, 0x0000    // RESERVED_MFR_385C
REG= 0x385E, 0x0000    // RESERVED_MFR_385E
REG= 0x3860, 0x0000    // RESERVED_MFR_3860
REG= 0x3862, 0x0000    // RESERVED_MFR_3862
REG= 0x3864, 0x0000    // RESERVED_MFR_3864
REG= 0x3866, 0x0000    // RESERVED_MFR_3866
REG= 0x3868, 0x0000    // RESERVED_MFR_3868
REG= 0x386A, 0x0000    // RESERVED_MFR_386A
REG= 0x386C, 0x0000    // RESERVED_MFR_386C
```

```
REG= 0x386E, 0x0000    // RESERVED_MFR_386E
REG= 0x3870, 0x0000    // RESERVED_MFR_3870
REG= 0x3872, 0x0000    // RESERVED_MFR_3872
REG= 0x3874, 0x0000    // RESERVED_MFR_3874
REG= 0x3876, 0x0000    // RESERVED_MFR_3876
REG= 0x3878, 0x0000    // RESERVED_MFR_3878
REG= 0x387A, 0x0000    // RESERVED_MFR_387A
REG= 0x387C, 0x0000    // RESERVED_MFR_387C
REG= 0x387E, 0x0000    // RESERVED_MFR_387E
REG= 0x3880, 0x0000    // RESERVED_MFR_3880
REG= 0x3882, 0x0000    // RESERVED_MFR_3882
REG= 0x3884, 0x0000    // RESERVED_MFR_3884
REG= 0x3886, 0x0000    // RESERVED_MFR_3886
REG= 0x3888, 0x0000    // RESERVED_MFR_3888
REG= 0x388A, 0x0000    // RESERVED_MFR_388A
REG= 0x388C, 0x0000    // RESERVED_MFR_388C
REG= 0x388E, 0x0000    // RESERVED_MFR_388E
REG= 0x3890, 0x0000    // RESERVED_MFR_3890
REG= 0x3892, 0x0000    // RESERVED_MFR_3892
REG= 0x3894, 0x0000    // RESERVED_MFR_3894
REG= 0x3896, 0x0000    // RESERVED_MFR_3896
REG= 0x3898, 0x0000    // RESERVED_MFR_3898
REG= 0x389A, 0x0000    // RESERVED_MFR_389A
REG= 0x389C, 0x0000    // RESERVED_MFR_389C
```

```
REG= 0x389E, 0x0000    // RESERVED_MFR_389E
REG= 0x38A0, 0x0000    // RESERVED_MFR_38A0
REG= 0x38A2, 0x0000    // RESERVED_MFR_38A2
REG= 0x38A4, 0x0000    // RESERVED_MFR_38A4
REG= 0x38A6, 0x0000    // RESERVED_MFR_38A6
REG= 0x38A8, 0x0000    // RESERVED_MFR_38A8
REG= 0x38AA, 0x0000    // RESERVED_MFR_38AA
REG= 0x38AC, 0x0000    // RESERVED_MFR_38AC
REG= 0x38AE, 0x0000    // RESERVED_MFR_38AE
REG= 0x38B0, 0x0000    // RESERVED_MFR_38B0
REG= 0x38B2, 0x0000    // RESERVED_MFR_38B2
REG= 0x38B4, 0x0000    // RESERVED_MFR_38B4
REG= 0x38B6, 0x0000    // RESERVED_MFR_38B6
REG= 0x38B8, 0x0000    // RESERVED_MFR_38B8
REG= 0x38BA, 0x0000    // RESERVED_MFR_38BA
REG= 0x38BC, 0x0000    // RESERVED_MFR_38BC
REG= 0x38BE, 0x0000    // RESERVED_MFR_38BE
REG= 0x38C0, 0x0000    // RESERVED_MFR_38C0
REG= 0x38C2, 0x0000    // RESERVED_MFR_38C2
REG= 0x38C4, 0x0000    // RESERVED_MFR_38C4
REG= 0x38C6, 0x0000    // RESERVED_MFR_38C6
REG= 0x38C8, 0x0000    // RESERVED_MFR_38C8
REG= 0x38CA, 0x0000    // RESERVED_MFR_38CA
REG= 0x38CC, 0x0000    // RESERVED_MFR_38CC
```

```
REG= 0x38CE, 0x0000    // RESERVED_MFR_38CE
REG= 0x38D0, 0x0000    // RESERVED_MFR_38D0
REG= 0x38D2, 0x0000    // RESERVED_MFR_38D2
REG= 0x38D4, 0x0000    // RESERVED_MFR_38D4
REG= 0x38D6, 0x0000    // RESERVED_MFR_38D6
REG= 0x38D8, 0x0000    // RESERVED_MFR_38D8
REG= 0x38DA, 0x0000    // RESERVED_MFR_38DA
REG= 0x38DC, 0x0000    // RESERVED_MFR_38DC
REG= 0x38DE, 0x0000    // RESERVED_MFR_38DE
REG= 0x38E0, 0x0000    // RESERVED_MFR_38E0
REG= 0x38E2, 0x0000    // RESERVED_MFR_38E2
REG= 0x38E4, 0x0000    // RESERVED_MFR_38E4
REG= 0x38E6, 0x0000    // RESERVED_MFR_38E6
REG= 0x38E8, 0x0000    // RESERVED_MFR_38E8
REG= 0x38EA, 0x0000    // RESERVED_MFR_38EA
REG= 0x38EC, 0x0000    // RESERVED_MFR_38EC
REG= 0x38EE, 0x0000    // RESERVED_MFR_38EE
REG= 0x38F0, 0x0000    // RESERVED_MFR_38F0
REG= 0x38F2, 0x0000    // RESERVED_MFR_38F2
REG= 0x38F4, 0x0000    // RESERVED_MFR_38F4
REG= 0x38F6, 0x0000    // RESERVED_MFR_38F6
REG= 0x38F8, 0x0000    // RESERVED_MFR_38F8
REG= 0x38FA, 0x0000    // RESERVED_MFR_38FA
REG= 0x38FC, 0x0000    // RESERVED_MFR_38FC
```

```
REG= 0x38FE, 0x0000     // RESERVED_MFR_38FE
REG= 0x3900, 0x0000     // RESERVED_MFR_3900
REG= 0x3902, 0x0000     // RESERVED_MFR_3902
REG= 0x3904, 0x0000     // RESERVED_MFR_3904
REG= 0x3906, 0x0000     // RESERVED_MFR_3906
REG= 0x3908, 0x0000     // RESERVED_MFR_3908
REG= 0x390A, 0x0000     // RESERVED_MFR_390A
REG= 0x390C, 0x0000     // RESERVED_MFR_390C
REG= 0x390E, 0x0000     // RESERVED_MFR_390E
REG= 0x3910, 0x0000     // RESERVED_MFR_3910
REG= 0x3912, 0x0000     // RESERVED_MFR_3912
REG= 0x3914, 0x0000     // RESERVED_MFR_3914
REG= 0x3916, 0x0000     // RESERVED_MFR_3916
REG= 0x3918, 0x0000     // RESERVED_MFR_3918
REG= 0x391A, 0x0000     // RESERVED_MFR_391A
REG= 0x391C, 0x0000     // RESERVED_MFR_391C
REG= 0x391E, 0x0000     // RESERVED_MFR_391E
REG= 0x3920, 0x0000     // RESERVED_MFR_3920
REG= 0x3922, 0x0000     // RESERVED_MFR_3922
REG= 0x3924, 0x0000     // RESERVED_MFR_3924
REG= 0x3926, 0x0000     // RESERVED_MFR_3926
REG= 0x3928, 0x0000     // RESERVED_MFR_3928
REG= 0x392A, 0x0000     // RESERVED_MFR_392A
REG= 0x392C, 0x0000     // RESERVED_MFR_392C
```

```
REG= 0x392E, 0x0000     // RESERVED_MFR_392E
REG= 0x3930, 0x0000     // RESERVED_MFR_3930
REG= 0x3932, 0x0000     // RESERVED_MFR_3932
REG= 0x3934, 0x0000     // RESERVED_MFR_3934
REG= 0x3936, 0x0000     // RESERVED_MFR_3936
REG= 0x3938, 0x0000     // RESERVED_MFR_3938
REG= 0x393A, 0x0000     // RESERVED_MFR_393A
REG= 0x393C, 0x0000     // RESERVED_MFR_393C
REG= 0x393E, 0x0000     // RESERVED_MFR_393E
REG= 0x3940, 0x0000     // RESERVED_MFR_3940
REG= 0x3942, 0x0000     // RESERVED_MFR_3942
REG= 0x3944, 0x0000     // RESERVED_MFR_3944
REG= 0x3946, 0x0000     // RESERVED_MFR_3946
REG= 0x3948, 0x0000     // RESERVED_MFR_3948
REG= 0x394A, 0x0000     // RESERVED_MFR_394A
REG= 0x394C, 0x0000     // RESERVED_MFR_394C
REG= 0x394E, 0x0000     // RESERVED_MFR_394E
REG= 0x3950, 0x0000     // RESERVED_MFR_3950
REG= 0x3952, 0x0000     // RESERVED_MFR_3952
REG= 0x3954, 0x0000     // RESERVED_MFR_3954
REG= 0x3956, 0x0000     // RESERVED_MFR_3956
REG= 0x3958, 0x0000     // RESERVED_MFR_3958
REG= 0x395A, 0x0000     // RESERVED_MFR_395A
REG= 0x395C, 0x0000     // RESERVED_MFR_395C
```

```
REG= 0x395E, 0x0000    // RESERVED_MFR_395E
REG= 0x3960, 0x0000    // RESERVED_MFR_3960
REG= 0x3962, 0x0000    // RESERVED_MFR_3962
REG= 0x3964, 0x0000    // RESERVED_MFR_3964
REG= 0x3966, 0x0000    // RESERVED_MFR_3966
REG= 0x3968, 0x0000    // RESERVED_MFR_3968
REG= 0x396A, 0x0000    // RESERVED_MFR_396A
REG= 0x396C, 0x0000    // RESERVED_MFR_396C
REG= 0x396E, 0x0000    // RESERVED_MFR_396E
REG= 0x3970, 0x0000    // RESERVED_MFR_3970
REG= 0x3972, 0x0000    // RESERVED_MFR_3972
REG= 0x3974, 0x0000    // RESERVED_MFR_3974
REG= 0x3976, 0x0000    // RESERVED_MFR_3976
REG= 0x3978, 0x0000    // RESERVED_MFR_3978
REG= 0x397A, 0x0000    // RESERVED_MFR_397A
REG= 0x397C, 0x0000    // RESERVED_MFR_397C
REG= 0x397E, 0x0000    // RESERVED_MFR_397E
REG= 0x3980, 0x0000    // RESERVED_MFR_3980
REG= 0x3982, 0x0000    // RESERVED_MFR_3982
REG= 0x3984, 0x0000    // RESERVED_MFR_3984
REG= 0x3986, 0x0000    // RESERVED_MFR_3986
REG= 0x3988, 0x0000    // RESERVED_MFR_3988
REG= 0x398A, 0x0000    // RESERVED_MFR_398A
REG= 0x398C, 0x0000    // RESERVED_MFR_398C
```

REG= 0x398E, 0x0000 // RESERVED_MFR_398E

REG= 0x3990, 0x0000 // RESERVED_MFR_3990

REG= 0x3992, 0x0000 // RESERVED_MFR_3992

REG= 0x3994, 0x0000 // RESERVED_MFR_3994

REG= 0x3996, 0x0000 // RESERVED_MFR_3996

REG= 0x3998, 0x0000 // RESERVED_MFR_3998

REG= 0x399A, 0x0000 // RESERVED_MFR_399A

REG= 0x399C, 0x0000 // RESERVED_MFR_399C

REG= 0x399E, 0x0000 // RESERVED_MFR_399E

REG= 0x39A0, 0x0000 // RESERVED_MFR_39A0

REG= 0x39A2, 0x0000 // RESERVED_MFR_39A2

REG= 0x39A4, 0x0000 // RESERVED_MFR_39A4

REG= 0x39A6, 0x0000 // RESERVED_MFR_39A6

REG= 0x39A8, 0x0000 // RESERVED_MFR_39A8

REG= 0x39AA, 0x0000 // RESERVED_MFR_39AA

REG= 0x39AC, 0x0000 // RESERVED_MFR_39AC

REG= 0x39AE, 0x0000 // RESERVED_MFR_39AE

REG= 0x39B0, 0x0000 // RESERVED_MFR_39B0

REG= 0x39B2, 0x0000 // RESERVED_MFR_39B2

REG= 0x39B4, 0x0000 // RESERVED_MFR_39B4

REG= 0x39B6, 0x0000 // RESERVED_MFR_39B6

REG= 0x39B8, 0x0000 // RESERVED_MFR_39B8

REG= 0x39BA, 0x0000 // RESERVED_MFR_39BA

REG= 0x39BC, 0x0000 // RESERVED_MFR_39BC

REG= 0x39BE, 0x0000 // RESERVED_MFR_39BE

REG= 0x39C0, 0x0000 // RESERVED_MFR_39C0

REG= 0x39C2, 0x0000 // RESERVED_MFR_39C2

REG= 0x39C4, 0x0000 // RESERVED_MFR_39C4

REG= 0x39C6, 0x0000 // RESERVED_MFR_39C6

REG= 0x39C8, 0x0000 // RESERVED_MFR_39C8

REG= 0x39CA, 0x0000 // RESERVED_MFR_39CA

REG= 0x39CC, 0x0000 // RESERVED_MFR_39CC

REG= 0x39CE, 0x0000 // RESERVED_MFR_39CE

REG= 0x39D0, 0x0000 // RESERVED_MFR_39D0

REG= 0x39D2, 0x0000 // RESERVED_MFR_39D2

REG= 0x39D4, 0x0000 // RESERVED_MFR_39D4

REG= 0x39D6, 0x0000 // RESERVED_MFR_39D6

REG= 0x39D8, 0x0000 // RESERVED_MFR_39D8

REG= 0x39DA, 0x0000 // RESERVED_MFR_39DA

REG= 0x39DC, 0x0000 // RESERVED_MFR_39DC

REG= 0x39DE, 0x0000 // RESERVED_MFR_39DE

REG= 0x39E0, 0x0000 // RESERVED_MFR_39E0

REG= 0x39E2, 0x0000 // RESERVED_MFR_39E2

REG= 0x39E4, 0x0000 // RESERVED_MFR_39E4

REG= 0x39E6, 0x0000 // RESERVED_MFR_39E6

REG= 0x39E8, 0x0000 // RESERVED_MFR_39E8

REG= 0x39EA, 0x0000 // RESERVED_MFR_39EA

REG= 0x39EC, 0x0000 // RESERVED_MFR_39EC

```
REG= 0x39EE, 0x0000    // RESERVED_MFR_39EE
REG= 0x39F0, 0x0000    // RESERVED_MFR_39F0
REG= 0x39F2, 0x0000    // RESERVED_MFR_39F2
REG= 0x39F4, 0x0000    // RESERVED_MFR_39F4
REG= 0x39F6, 0x0000    // RESERVED_MFR_39F6
REG= 0x39F8, 0x0000    // RESERVED_MFR_39F8
REG= 0x39FA, 0x0000    // RESERVED_MFR_39FA
REG= 0x39FC, 0x0000    // RESERVED_MFR_39FC
REG= 0x39FE, 0x0000    // RESERVED_MFR_39FE
REG= 0x3E00, 0x0010    // RESERVED_MFR_3E00
REG= 0x3E02, 0xDE02    // RESERVED_MFR_3E02
REG= 0x3E04, 0x00FF    // RESERVED_MFR_3E04
REG= 0x3E06, 0x00FF    // RESERVED_MFR_3E06
REG= 0x3E08, 0xDC20    // RESERVED_MFR_3E08
REG= 0x3E0A, 0xDC06    // RESERVED_MFR_3E0A
REG= 0x3E0C, 0x3824    // RESERVED_MFR_3E0C
REG= 0x3E0E, 0x3425    // RESERVED_MFR_3E0E
REG= 0x3E10, 0x3622    // RESERVED_MFR_3E10
REG= 0x3E12, 0x0000    // RESERVED_MFR_3E12
REG= 0x3E14, 0xDA80    // RESERVED_MFR_3E14
REG= 0x3E16, 0x7C22    // RESERVED_MFR_3E16
REG= 0x3E18, 0x9C7E    // RESERVED_MFR_3E18
REG= 0x3E1A, 0x9980    // RESERVED_MFR_3E1A
REG= 0x3E1C, 0x9A7C    // RESERVED_MFR_3E1C
```

```
REG= 0x3E1E, 0x0000    // RESERVED_MFR_3E1E
REG= 0x3E20, 0xDC06    // RESERVED_MFR_3E20
REG= 0x3E22, 0x00FF    // RESERVED_MFR_3E22
REG= 0x3E24, 0xDC02    // RESERVED_MFR_3E24
REG= 0x3E26, 0xDC02    // RESERVED_MFR_3E26
REG= 0x3E28, 0xDC1E    // RESERVED_MFR_3E28
REG= 0x3E2A, 0xEE02    // RESERVED_MFR_3E2A
REG= 0x3E2C, 0x00FF    // RESERVED_MFR_3E2C
REG= 0x3E2E, 0x00FF    // RESERVED_MFR_3E2E
REG= 0x3E30, 0x00E8    // RESERVED_MFR_3E30
REG= 0x3E32, 0x52F0    // RESERVED_MFR_3E32
REG= 0x3E34, 0x0000    // RESERVED_MFR_3E34
REG= 0x3E36, 0x0000    // RESERVED_MFR_3E36
REG= 0x3E38, 0xDE04    // RESERVED_MFR_3E38
REG= 0x3E3A, 0xFF00    // RESERVED_MFR_3E3A
REG= 0x3E3C, 0x0000    // RESERVED_MFR_3E3C
REG= 0x3E3E, 0xDE02    // RESERVED_MFR_3E3E
REG= 0x3E40, 0xDC05    // RESERVED_MFR_3E40
REG= 0x3E42, 0x6E22    // RESERVED_MFR_3E42
REG= 0x3E44, 0xDC22    // RESERVED_MFR_3E44
REG= 0x3E46, 0xFF00    // RESERVED_MFR_3E46
REG= 0x3E48, 0xDC02    // RESERVED_MFR_3E48
REG= 0x3E4A, 0xDC02    // RESERVED_MFR_3E4A
REG= 0x3E4C, 0xDC1E    // RESERVED_MFR_3E4C
```

```
REG= 0x3E4E, 0xDC02    // RESERVED_MFR_3E4E

REG= 0x3E50, 0xDC1E    // RESERVED_MFR_3E50

REG= 0x3E52, 0xFF01    // RESERVED_MFR_3E52

REG= 0x3E54, 0x3222    // RESERVED_MFR_3E54

REG= 0x3E56, 0x00FF    // RESERVED_MFR_3E56

REG= 0x3E58, 0xDE04    // RESERVED_MFR_3E58

REG= 0x3E90, 0x5203    // RESERVED_MFR_3E90

REG= 0x3E92, 0x5005    // RESERVED_MFR_3E92

REG= 0x3E94, 0x4C06    // RESERVED_MFR_3E94

REG= 0x3E96, 0x4806    // RESERVED_MFR_3E96

REG= 0x3E98, 0x4607    // RESERVED_MFR_3E98

REG= 0x3E9A, 0x4A05    // RESERVED_MFR_3E9A

REG= 0x3E9C, 0x0000    // RESERVED_MFR_3E9C

REG= 0x3E9E, 0x4E04    // RESERVED_MFR_3E9E

REG= 0x3EA0, 0x0000    // RESERVED_MFR_3EA0

REG= 0x3EA2, 0x5200    // RESERVED_MFR_3EA2

REG= 0x3EB0, 0x0507    // RESERVED_MFR_3EB0

REG= 0x3EB2, 0x0608    // RESERVED_MFR_3EB2

REG= 0x3EB4, 0x97C7    // RESERVED_MFR_3EB4

REG= 0x3EB6, 0x97C6    // RESERVED_MFR_3EB6

REG= 0x3EB8, 0x0502    // RESERVED_MFR_3EB8

REG= 0x3ECC, 0x0FE4    // RESERVED_MFR_3ECC

REG= 0x3ECE, 0x1019    // RESERVED_MFR_3ECE

REG= 0x3ED0, 0x1B24    // RESERVED_MFR_3ED0
```

```
REG= 0x3ED2, 0xA660   // RESERVED_MFR_3ED2
REG= 0x3ED4, 0xF998   // RESERVED_MFR_3ED4
REG= 0x3ED6, 0x9789   // RESERVED_MFR_3ED6
REG= 0x3ED8, 0x5803   // RESERVED_MFR_3ED8
REG= 0x3EDA, 0xD9C3   // RESERVED_MFR_3EDA
REG= 0x3EDC, 0xD5E4   // RESERVED_MFR_3EDC
REG= 0x3EDE, 0xE41A   // RESERVED_MFR_3EDE
REG= 0x3EE0, 0xA43F   // RESERVED_MFR_3EE0
REG= 0x3EE2, 0xA4BF   // RESERVED_MFR_3EE2
REG= 0x3EE4, 0xE4E4   // RESERVED_MFR_3EE4
REG= 0x3EE6, 0x4540   // RESERVED_MFR_3EE6
REG= 0x3EE8, 0x0001   // RESERVED_MFR_3EE8
REG= 0x3EEA, 0x5500   // RESERVED_MFR_3EEA
REG= 0x3EEC, 0x1C21   // RESERVED_MFR_3EEC
REG= 0x3EEE, 0x1212   // RESERVED_MFR_3EEE
REG= 0x3EF0, 0x1212   // RESERVED_MFR_3EF0
```

APPENDIX B

Captured Data Image Received from Small Opening Lensless Camera

Captured data image received from small opening lensless camera

```
//  Captured 15:47:52 - Wednesday, June 11, 2014
//
//  Sensor info:
//          Width         = 1920
//          Height        = 1082
//          Image Format   = Bayer 12
//          Subformat      = GRBG
//          Sensor output  = 12 bits per pixel
//
//  .RAW file info:
//          stored as      = 16 bits unsigned short
//          valid data     = xxxxBA98 76543210
//          Endianess      = little
//
//  .BMP file info (24-bit windows bitmap file):
//          Width         = 1920
//          Height        = 1082
//
//  Application version info:
//          Application Name = C:\Aptina Imaging\DevWare.exe
//          Application Version = 4.5.23.39222
//          Application Type = 4.5.23_Release
```

```
//	Application Date = 05/05/2014
//
// Camera info:
//	Product ID = 0x100D Version = 0xC1
//	Product Name = Aptina Imaging DEMO2X
//	Firmware Version = D.2A
//	Transport Name = USB 2.0
//	Chip 0 Name = DEMO2X C1
//		CHIP_VERSION_REG = 0x00C1
//	Chip 1 Name = High Speed Serial Adapter
//		VERSION = 0x0131
//		CHIP_VERSION_REG = 0x00CD
//		DATESTAMPREG = 0x304A
//		TIMESTAMPREG = 0x1016
//	Sensor Name = A-10030
//	Sensor Part Number = MT9J003
//	Sensor Version = REV3
//	Sensor Filename = C:\Aptina Imaging\sensor_data\MT9J003-REV3.xsdat
//
// Sensor Fuse info:
//	FuseID: D652768C5758C679
//	Revision: 2
//	Silicon Option: --
//	Customer Revision: 0x32
```

```
//
// Windows OS info:
//      Display resolution = 1600x900 at 32bpp
//      OS Versioninfo = (6, 1, 7600, 2, , 0, 0)
//      Microsoft Windows Windows 7
//
//
// Processor info:
//      2095 MHz
//      GenuineIntel
//      Intel(R) Core(TM) i7-3612QM CPU @ 2.10GHz
//      50 percent of memory is in use.
//      Memory 8063 MB (total)
//      Memory 4007 MB (available)
//
// Display Devices:
//      Device 0: Intel(R) HD Graphics 4000
//
// USB Host Controllers:
//      Service: usbehci
//              Driver File: C:\Windows\system32\DRIVERS\usbehci.sys
//              File Version: 6.1.7601.18328
//              Device Desc 0:
PCI\VEN_8086&DEV_1E2D&SUBSYS_05641028&REV_04\3&11583659&0&D0
//              Device Desc 1:
```

PCI\VEN_8086&DEV_1E26&SUBSYS_05641028&REV_04\3&11583659&0&E8

//

// Camera Driver Info:

// C:\Windows\System32\Drivers\USB64W7.SYS (3.4.1.20) - VendorID: 0x20FB (Aptina Imaging)

[ColorPipe State]

STATE= Display Zoom Percent, 33

STATE= Master Clock, 71000000

STATE= Update Sensor FPS, 1

STATE= Allow Update Sensor FPS, 1

STATE= Filter, 0

STATE= X Offset, 0

STATE= Y Offset, 0

STATE= Auto Offset, 1

STATE= CFA Pattern, 1

STATE= Gb-B Swap, 0

STATE= RGBC BiWindow, 2

STATE= Monochrome, 0

STATE= Bayer Quadrant, 0

STATE= Byte Swap, 0

STATE= RedBlue Swap, 0

STATE= True Black Scale, 4096

STATE= True Black Level, 40

STATE= True Black Enable, 2

STATE= Auto Luma Range, 1

STATE= Luma Lo, 0

STATE= Luma Hi, 255

STATE= Unswizzle Mode, 3

STATE= Swap 12-bit LSBs, 0

STATE= Column Repeat, 0

STATE= Orientation, 0

STATE= Deinterlace Mode, 3

STATE= Descramble Mode, 1

STATE= Special Pixel Mode, 0

STATE= Active Area Crop, 0

STATE= Output Channel, 0

STATE= Output BwColor, 0

STATE= X Bin, 1

STATE= Y Bin, 1

STATE= DVS Split Screen, 0

STATE= Stereo Merge, 0

STATE= Stereo Shift X, 0

STATE= Stereo Shift Y, 0

STATE= Stereo Colwise, 0

STATE= sRGB Color Standard, 0

STATE= Color Correction, 1

STATE= Gamma Correction, 45

STATE= Black Correct, 5

STATE= Saturation, 10

STATE= Contrast, 25

STATE= Aperture Enable, 1

STATE= Aperture, 4

STATE= Black CCM Kill Enable, 0

STATE= Black CCM Kill A, 240

STATE= Black CCM Kill B, 160

STATE= Black CCM Kill C, 80

STATE= Green Balance Enable, 0

STATE= Green Balance Apos, 128

STATE= Green Balance Bpos, 10

STATE= Green Balance Aneg, -128

STATE= Green Balance Bneg, 10

STATE= Auto Exposure, 0

STATE= Auto Exposure Target, 50

STATE= Auto Exposure Stability, 6

STATE= Auto Exposure Speed, 30

STATE= Auto Exposure Minimum FPS, 30

STATE= Auto Exposure Flicker Filter, 0

STATE= Auto Exposure Soft Limit, 33

STATE= Auto Exposure Soft Gain Limit, 40

STATE= Auto Exposure Gain Limit, 317

STATE= Auto Exposure Minimum Global Gain, 10

STATE= Auto Exposure Global Gain Limit, 10

STATE= Auto Exposure Software Gain Limit, 10

STATE= Auto Exposure Freeze Gains, 1

STATE= Auto Exposure Fade Saturation, 1

STATE= Auto Exposure Fade Aperture, 1

STATE= Auto Exposure Fade Target, 1

STATE= Auto Exposure Inner Zone, 50

STATE= Auto Exposure Outer Zone, 50

STATE= HDR AE Mode, 0

STATE= Software Gain, 1000

STATE= Mechanical Shutter Same, 1

STATE= Mechanical Shutter Time, 33333

STATE= Mechanical Shutter Delay, 0

STATE= Trigger Width, 0

STATE= White Balance, 1

STATE= WB Speed, 30

STATE= WB Adjust Gains, 0

STATE= WB Manual Position, 0

STATE= WB Manual RedGreen, 89

STATE= WB Manual BlueGreen, 198

STATE= WB Interpolate Saturation, 1

STATE= WB Normalize Matrix, 1

STATE= AWB Weight Map Method, 2

STATE= AWB Weight Map X Scale, 128

STATE= AWB Weight Map Y Scale, 256

STATE= AWB Weight Map X Shift, 32

STATE= AWB Weight Map Y Shift, 8

STATE= AWB Weight Map X Center, 1014

STATE= AWB Weight Map Y Center, 1009

STATE= AWB Weight Map Angle Sin, 48

STATE= AWB Weight Map Angle Cos, 43

STATE= AWB Weight Map Luma Low, 4

STATE= AWB Weight Map Luma High, 251

STATE= Minimum Gain, 1000

STATE= Show Min Gain As 1, 1

STATE= Default Relative Red Gain, 1000

STATE= Default Relative Blue Gain, 1570

STATE= Relative Red Gain, 1750

STATE= Relative Blue Gain, 1297

STATE= Lens Correction Enable, 0

STATE= Lens Correction Falloff, 100

STATE= Lens Correction Falloff R, 100

STATE= Lens Correction Falloff G1, 100

STATE= Lens Correction Falloff G2, 100

STATE= Lens Correction Falloff B, 100

STATE= Lens Correction Overlay, 0

STATE= Lens Correction Center X, 1920

STATE= Lens Correction Center Y, 1374

STATE= Lens Correction Coeff Prec, 16

STATE= Lens Correction Lens Radius, 0

STATE= Lens Correction Luma Only, 0

STATE= Lens Center Red X, 1920

STATE= Lens Center Red Y, 1374

STATE= Lens Center Green1 X, 1920

STATE= Lens Center Green1 Y, 1374

STATE= Lens Center Green2 X, 1920

STATE= Lens Center Green2 Y, 1374

STATE= Lens Center Blue X, 1920

STATE= Lens Center Blue Y, 1374

STATE= Lens Center Overlay, 0

STATE= Lens Sim Sensor, 0

STATE= Lens Sim Sensor Rev, 0

STATE= Lens Sim Enable Pwq, 0

STATE= Lens Sim Enable Poly, 0

STATE= Noise Removal, 45

STATE= Noise Removal Level, 2

STATE= Noise Removal Depth, 6

STATE= Noise Removal K1, 2000

STATE= Noise Removal K2, 1800

STATE= Noise Removal K3, 1000

STATE= Noise Removal Edges, 1

STATE= Noise Removal Kernel, 0

STATE= Optical Black, 0

STATE= Optical Black Row Filter, 1

STATE= Optical Black 1 Row Start, 0

STATE= Optical Black 1 Row End, 0

STATE= Optical Black 2 Row Start, 0

STATE= Optical Black 2 Row End, 0

STATE= Optical Black 1 Column Start, 0

STATE= Optical Black 1 Column End, 0

STATE= Optical Black 2 Column Start, 0

STATE= Optical Black 2 Column End, 0

STATE= ALTM Enable, 0

STATE= Defect Enable, 1

STATE= Defect Max, 10000

STATE= Defect Auto Defect Correction, 0

STATE= Flash Lamp, 0

STATE= Still Global Reset, 0

STATE= Global Reset Bulb, 0

STATE= Continuous GRR, 0

STATE= Num Capture Frames, 1

STATE= Still Mode, 0

STATE= Still Hold, 1

STATE= Still Capture Average, 0

STATE= Still Capture Timeout, 5

STATE= Still HalfPress, 0

STATE= Delay before snap, 0

STATE= Save 24bpp BMP, 1

STATE= Save RAW, 1

STATE= Save TXT, 1

STATE= Save HEX, 0

STATE= Save ITX, 0

STATE= Save CCR, 0

STATE= Save DXR, 0

STATE= Save 48bpp COLOR TIFF, 0

STATE= Save JPEG, 0

STATE= Save RAW JPEG, 0

STATE= Save BMP Info, 0

STATE= JPEG Quality (1-100), 98

STATE= Save RAW PNG, 0

STATE= Save PNG, 0

STATE= Save DNG, 0

STATE= Save SS, 0

STATE= Save Selection Rectangle, 0

STATE= ICC Profile, 0

STATE= Video Screen Capture, 1

STATE= VidCap Play FPS, 15000

STATE= VidCap Auto Play FPS, 1

STATE= VidCap Format, 0

STATE= VidCap FileName Increment, 1

STATE= RAM Capture, 0

STATE= RAM Capture MB, 128

STATE= RAM Capture Cycle, 1

STATE= Preview Recording, 1

STATE= WB Xenon Red Gain, 1024

STATE= WB Xenon Blue Gain, 1024

STATE= WB Led Red Gain, 1024

STATE= WB Led Blue Gain, 1024

STATE= MAE Overlay, 0

STATE= Noise Image Type, 0

STATE= Noise Frames, 50

STATE= Noise Defects, 0

STATE= Strip FSP, 1

STATE= Check Thumbnail Table, 0

STATE= CRA Overlay, 0

STATE= Allow FAR Access, 1

STATE= Pure Raw, 0

STATE= Sensor Orientation, 0

STATE= Clarity6, 3

STATE= Clarity7, 1

STATE= Dynamic Range LSB, 0

STATE= Dynamic Range MSB, 31

STATE= AI Repack, 1

STATE= AR1820 REV12 GHR Workaround, 0

STATE= Deinterleave HDR, 3

STATE= HDR AE Histogram High Percent, 0.99

STATE= HDR AE Histogram High Target, 0.75

STATE= HDR AE Scene Brightness Offset, 8

STATE= HDR AE Max Percent Adjust, 0

STATE= HDR AE Luma Target, 40 80 200 600 1000 1400 1800 2200

STATE= AWB Incandescent, 1.886 -1.248 0.362 -0.128 1.329 -0.201 -0.082 -0.775 1.857

STATE= AWB Sun, 2.036 -1.055 0.019 -0.086 1.351 -0.265 0.007 -0.551 1.544

STATE= AWB Incandescent Gain, 0.896 1.987

STATE= AWB Sun Gain, 1.691 0.957

STATE= WB Custom, 1 0 0 0 1 0 0 0 1

STATE= WB Custom Xenon, 1 0 0 0 1 0 0 0 1

STATE= WB Custom Led, 1 0 0 0 1 0 0 0 1

STATE= AWB Weight Map, 0 3 8224 4096 1 273 4866 4368 34 8995 8739 12816 308 17204 12612 17168 546 21349 4915 13361 291 21333 8241 12832 3 8755 4610 4640 17 275 0 0

STATE= Optical Black Level Rect, 0 0 0 0

STATE= Active Area Rect, -1 -1 -1 -1

STATE= Lens Curve Red,
C04192013743F8000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000

STATE= Lens Curve Green1,
C04192013743F8000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000

STATE= Lens Curve Green2,
C04192013743F8000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000

STATE= Lens Curve Blue,
C04192013743F8000000000000000000000000000000000000000000000000000000

0000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000

STATE= RGBC Sigma_S, 1

STATE= RGBC Sigma_I, 0.005

STATE= RGBC Smooth_Th, 0.1

STATE= ALTM Peak Percent, 0.99

STATE= ALTM Sharp S, 1

STATE= HW Gain, 0.5

STATE= HW Global Gain, 1

STATE= HW Red Gain, 0.5

STATE= HW Green1 Gain, 0.5

STATE= HW Green2 Gain, 0.5

STATE= HW Blue Gain, 0.5

STATE= HW Exposure Time, 0.749994

STATE= VidCap File, C:\Users\Aleksandar\Documents\Aptina Imaging\video.avi

STATE= Clarity2, 1

STATE= Clarity3, 0.04

STATE= Clarity4, 0

STATE= Clarity5, 0.34

STATE= Chroma Filter Absolute Add Corners, 0

STATE= Clarity23, 1 7 0

STATE= Clarity24, 24aff 1249e

STATE= Clarity8, 0.028

STATE= Clarity9, 0.12

STATE= Clarity25, 2 2

STATE= Clarity11, 0

STATE= Clarity12, 0.117

STATE= Clarity10, 0.05

STATE= Clarity22, 0.025

STATE= Clarity13, 0.04

STATE= Clarity14, 0.2

STATE= Clarity15, 8

STATE= Black Balance, 0 0 0 0

STATE= Clarity18, 1

STATE= Clarity17, 2

STATE= Clarity20, 0.24 0.17

STATE= Clarity19, 0.1 0.1

STATE= Clarity21, 0.05

STATE= Clarity16, 0.01

STATE= Clarity26, 0.1

STATE= Clarity27, 1

STATE= Post Gain, 1

STATE= NIR Factor Red, 1

STATE= NIR Factor Green, 1

STATE= NIR Factor Blue, 1

STATE= NIR Coeff Red, 0

STATE= NIR Coeff Green, 0

STATE= NIR Coeff Blue, 0

STATE= NIR WB Custom, 1 0 0 0 1 0 0 0 1

STATE= NIR WB Gains, 1 1 1

STATE= NIR Global Gain, 1

[Raw Image Format]

IMAGE= 1920, 1082, BAYER-12

[Register State]

```
REG= 0x0000, 0x2C01    // CHIP_VERSION_REG
REG= 0x0002, 0x20      // REVISION_NUMBER
REG= 0x0003, 0x06      // MANUFACTURER_ID
REG= 0x0004, 0x0A      // SMIA_VERSION
REG= 0x0005, 0xB9      // FRAME_COUNT
REG= 0x0006, 0x00      // PIXEL_ORDER
REG= 0x0008, 0x0028    // DATA_PEDESTAL
REG= 0x0040, 0x01      // FRAME_FORMAT_MODEL_TYPE
REG= 0x0041, 0x12      // FRAME_FORMAT_MODEL_SUBTYPE
REG= 0x0042, 0x5780    // FRAME_FORMAT_DESCRIPTOR_0
REG= 0x0044, 0x1002    // FRAME_FORMAT_DESCRIPTOR_1
REG= 0x0046, 0x5438    // FRAME_FORMAT_DESCRIPTOR_2
REG= 0x0048, 0x0000    // FRAME_FORMAT_DESCRIPTOR_3
REG= 0x004A, 0x0000    // FRAME_FORMAT_DESCRIPTOR_4
REG= 0x004C, 0x0000    // FRAME_FORMAT_DESCRIPTOR_5
REG= 0x004E, 0x0000    // FRAME_FORMAT_DESCRIPTOR_6
REG= 0x0050, 0x0000    // FRAME_FORMAT_DESCRIPTOR_7
```

```
REG= 0x0052, 0x0000    // FRAME_FORMAT_DESCRIPTOR_8
REG= 0x0054, 0x0000    // FRAME_FORMAT_DESCRIPTOR_9
REG= 0x0056, 0x0000    // FRAME_FORMAT_DESCRIPTOR_10
REG= 0x0058, 0x0000    // FRAME_FORMAT_DESCRIPTOR_11
REG= 0x005A, 0x0000    // FRAME_FORMAT_DESCRIPTOR_12
REG= 0x005C, 0x0000    // FRAME_FORMAT_DESCRIPTOR_13
REG= 0x005E, 0x0000    // FRAME_FORMAT_DESCRIPTOR_14
REG= 0x0080, 0x0001    // ANALOGUE_GAIN_CAPABILITY
REG= 0x0084, 0x0008    // ANALOGUE_GAIN_CODE_MIN
REG= 0x0086, 0x00FF    // ANALOGUE_GAIN_CODE_MAX
REG= 0x0088, 0x0001    // ANALOGUE_GAIN_CODE_STEP
REG= 0x008A, 0x0000    // ANALOGUE_GAIN_TYPE
REG= 0x008C, 0x0001    // ANALOGUE_GAIN_M0
REG= 0x008E, 0x0000    // ANALOGUE_GAIN_C0
REG= 0x0090, 0x0000    // ANALOGUE_GAIN_M1
REG= 0x0092, 0x0008    // ANALOGUE_GAIN_C1
REG= 0x00C0, 0x01      // DATA_FORMAT_MODEL_TYPE
REG= 0x00C1, 0x05      // DATA_FORMAT_MODEL_SUBTYPE
REG= 0x00C2, 0x0A0A    // DATA_FORMAT_DESCRIPTOR_0
REG= 0x00C4, 0x0808    // DATA_FORMAT_DESCRIPTOR_1
REG= 0x00C6, 0x0A08    // DATA_FORMAT_DESCRIPTOR_2
REG= 0x00C8, 0x0C0C    // DATA_FORMAT_DESCRIPTOR_3
REG= 0x00CA, 0x0C08    // DATA_FORMAT_DESCRIPTOR_4
REG= 0x00CC, 0x0000    // DATA_FORMAT_DESCRIPTOR_5
```

```
REG= 0x00CE, 0x0000    // DATA_FORMAT_DESCRIPTOR_6
REG= 0x0100, 0x01      // MODE_SELECT
REG= 0x0101, 0x00      // IMAGE_ORIENTATION
REG= 0x0103, 0x00      // SOFTWARE_RESET
REG= 0x0104, 0x00      // GROUPED_PARAMETER_HOLD
REG= 0x0105, 0x00      // MASK_CORRUPTED_FRAMES
REG= 0x0110, 0x00      // CCP2_CHANNEL_IDENTIFIER
REG= 0x0111, 0x01      // CCP2_SIGNALLING_MODE
REG= 0x0112, 0x0C0C    // CCP_DATA_FORMAT
REG= 0x0120, 0x00      // GAIN_MODE
REG= 0x0200, 0x03F2    // FINE_INTEGRATION_TIME
REG= 0x0202, 0x5A70    // COARSE_INTEGRATION_TIME
REG= 0x0204, 0x0004    // ANALOGUE_GAIN_CODE_GLOBAL
REG= 0x0206, 0x0004    // ANALOGUE_GAIN_CODE_GREENR
REG= 0x0208, 0x0004    // ANALOGUE_GAIN_CODE_RED
REG= 0x020A, 0x0004    // ANALOGUE_GAIN_CODE_BLUE
REG= 0x020C, 0x0004    // ANALOGUE_GAIN_CODE_GREENB
REG= 0x020E, 0x0100    // DIGITAL_GAIN_GREENR
REG= 0x0210, 0x0100    // DIGITAL_GAIN_RED
REG= 0x0212, 0x0100    // DIGITAL_GAIN_BLUE
REG= 0x0214, 0x0100    // DIGITAL_GAIN_GREENB
REG= 0x0300, 0x0003    // VT_PIX_CLK_DIV
REG= 0x0302, 0x0001    // VT_SYS_CLK_DIV
REG= 0x0304, 0x0003    // PRE_PLL_CLK_DIV
```

```
REG= 0x0306, 0x0030    // PLL_MULTIPLIER
REG= 0x0308, 0x000C    // OP_PIX_CLK_DIV
REG= 0x030A, 0x0001    // OP_SYS_CLK_DIV
REG= 0x0340, 0x048A    // FRAME_LENGTH_LINES
REG= 0x0342, 0x08FC    // LINE_LENGTH_PCK
REG= 0x0344, 0x0020    // X_ADDR_START
REG= 0x0346, 0x0128    // Y_ADDR_START
REG= 0x0348, 0x0F21    // X_ADDR_END
REG= 0x034A, 0x0995    // Y_ADDR_END
REG= 0x034C, 0x0780    // X_OUTPUT_SIZE
REG= 0x034E, 0x0438    // Y_OUTPUT_SIZE
REG= 0x0380, 0x0001    // X_EVEN_INC
REG= 0x0382, 0x0003    // X_ODD_INC
REG= 0x0384, 0x0001    // Y_EVEN_INC
REG= 0x0386, 0x0003    // Y_ODD_INC
REG= 0x0400, 0x0002    // SCALING_MODE
REG= 0x0402, 0x0000    // SPATIAL_SAMPLING
REG= 0x0404, 0x0010    // SCALE_M
REG= 0x0406, 0x0010    // SCALE_N
REG= 0x0500, 0x0001    // COMPRESSION_MODE
REG= 0x0600, 0x0000    // TEST_PATTERN_MODE
REG= 0x0602, 0x0000    // TEST_DATA_RED
REG= 0x0604, 0x0000    // TEST_DATA_GREENR
REG= 0x0606, 0x0000    // TEST_DATA_BLUE
```

```
REG= 0x0608, 0x0000    // TEST_DATA_GREENB
REG= 0x060A, 0x0000    // HORIZONTAL_CURSOR_WIDTH
REG= 0x060C, 0x0000    // HORIZONTAL_CURSOR_POSITION
REG= 0x060E, 0x0000    // VERTICAL_CURSOR_WIDTH
REG= 0x0610, 0x0000    // VERTICAL_CURSOR_POSITION
REG= 0x1000, 0x0001    // INTEGRATION_TIME_CAPABILITY
REG= 0x1004, 0x0000    // COARSE_INTEGRATION_TIME_MIN
REG= 0x1006, 0x0001    // COARSE_INTEGRATION_TIME_MAX_MARGIN
REG= 0x1008, 0x03F2    // FINE_INTEGRATION_TIME_MIN
REG= 0x100A, 0x027E    // FINE_INTEGRATION_TIME_MAX_MARGIN
REG= 0x1080, 0x0001    // DIGITAL_GAIN_CAPABILITY
REG= 0x1084, 0x0100    // DIGITAL_GAIN_MIN
REG= 0x1086, 0x0700    // DIGITAL_GAIN_MAX
REG= 0x1088, 0x0100    // DIGITAL_GAIN_STEP_SIZE
REG= 0x1100, 0x40000000        // MIN_EXT_CLK_FREQ_MHZ
REG= 0x1104, 0x42800000        // MAX_EXT_CLK_FREQ_MHZ
REG= 0x1108, 0x0001    // MIN_PRE_PLL_CLK_DIV
REG= 0x110A, 0x0040    // MAX_PRE_PLL_CLK_DIV
REG= 0x110C, 0x40800000        // MIN_PLL_IP_FREQ_MHZ
REG= 0x1110, 0x41C00000        // MAX_PLL_IP_FREQ_MHZ
REG= 0x1114, 0x0020    // MIN_PLL_MULTIPLIER
REG= 0x1116, 0x0180    // MAX_PLL_MULTIPLIER
REG= 0x1118, 0x43C00000        // MIN_PLL_OP_FREQ_MHZ
REG= 0x111C, 0x44400000        // MAX_PLL_OP_FREQ_MHZ
```

REG= 0x1120, 0x0001 // MIN_VT_SYS_CLK_DIV

REG= 0x1122, 0x0001 // MAX_VT_SYS_CLK_DIV

REG= 0x1124, 0x41C00000 // MIN_VT_SYS_CLK_FREQ_MHZ

REG= 0x1128, 0x44400000 // MAX_VT_SYS_CLK_FREQ_MHZ

REG= 0x112C, 0x4019999A // MIN_VT_PIX_CLK_FREQ_MHZ

REG= 0x1130, 0x42C00000 // MAX_VT_PIX_CLK_FREQ_MHZ

REG= 0x1134, 0x0004 // MIN_VT_PIX_CLK_DIV

REG= 0x1136, 0x0006 // MAX_VT_PIX_CLK_DIV

REG= 0x1140, 0x0091 // MIN_FRAME_LENGTH_LINES

REG= 0x1142, 0xFFFF // MAX_FRAME_LENGTH_LINES

REG= 0x1144, 0x0670 // MIN_LINE_LENGTH_PCK

REG= 0x1146, 0xFFFE // MAX_LINE_LENGTH_PCK

REG= 0x1148, 0x046E // MIN_LINE_BLANKING_PCK

REG= 0x114A, 0x008F // MIN_FRAME_BLANKING_LINES

REG= 0x1160, 0x0001 // MIN_OP_SYS_CLK_DIV

REG= 0x1162, 0x0001 // MAX_OP_SYS_CLK_DIV

REG= 0x1164, 0x4199999A // MIN_OP_SYS_CLK_FREQ_MHZ

REG= 0x1168, 0x44400000 // MAX_OP_SYS_CLK_FREQ_MHZ

REG= 0x116C, 0x0008 // MIN_OP_PIX_CLK_DIV

REG= 0x116E, 0x000C // MAX_OP_PIX_CLK_DIV

REG= 0x1170, 0x4019999A // MIN_OP_PIX_CLK_FREQ_MHZ

REG= 0x1174, 0x42C00000 // MAX_OP_PIX_CLK_FREQ_MHZ

REG= 0x1180, 0x0018 // X_ADDR_MIN

REG= 0x1182, 0x0000 // Y_ADDR_MIN

```
REG= 0x1184, 0x0F27    // X_ADDR_MAX

REG= 0x1186, 0x0ACB    // Y_ADDR_MAX

REG= 0x11C0, 0x0001    // MIN_EVEN_INC

REG= 0x11C2, 0x0001    // MAX_EVEN_INC

REG= 0x11C4, 0x0001    // MIN_ODD_INC

REG= 0x11C6, 0x0003    // MAX_ODD_INC

REG= 0x1200, 0x0002    // SCALING_CAPABILITY

REG= 0x1204, 0x0010    // SCALER_M_MIN

REG= 0x1206, 0x0080    // SCALER_M_MAX

REG= 0x1208, 0x0010    // SCALER_N_MIN

REG= 0x120A, 0x0010    // SCALER_N_MAX

REG= 0x1300, 0x0001    // COMPRESSION_CAPABILITY

REG= 0x1400, 0x0242    // MATRIX_ELEMENT_REDINRED

REG= 0x1402, 0xFF00    // MATRIX_ELEMENT_GREENINRED

REG= 0x1404, 0xFFBE    // MATRIX_ELEMENT_BLUEINRED

REG= 0x1406, 0xFFB4    // MATRIX_ELEMENT_REDINGREEN

REG= 0x1408, 0x0200    // MATRIX_ELEMENT_GREENINGREEN

REG= 0x140A, 0xFF4D    // MATRIX_ELEMENT_BLUEINGREEN

REG= 0x140C, 0xFFF1    // MATRIX_ELEMENT_REDINBLUE

REG= 0x140E, 0xFF34    // MATRIX_ELEMENT_GREENINBLUE

REG= 0x1410, 0x01DC    // MATRIX_ELEMENT_BLUEINBLUE

REG= 0x3000, 0x2C01    // MODEL_ID_

REG= 0x3002, 0x0128    // Y_ADDR_START_

REG= 0x3004, 0x0020    // X_ADDR_START_
```

```
REG= 0x3006, 0x0995    // Y_ADDR_END_
REG= 0x3008, 0x0F21    // X_ADDR_END_
REG= 0x300A, 0x048A    // FRAME_LENGTH_LINES_
REG= 0x300C, 0x08FC    // LINE_LENGTH_PCK_
REG= 0x3010, 0x009C    // FINE_CORRECTION
REG= 0x3012, 0x5A70    // COARSE_INTEGRATION_TIME_
REG= 0x3014, 0x03F2    // FINE_INTEGRATION_TIME_
REG= 0x3016, 0x0121    // ROW_SPEED
REG= 0x3018, 0x0000    // EXTRA_DELAY
REG= 0x301A, 0x001C    // RESET_REGISTER
REG= 0x301C, 0x01      // MODE_SELECT_
REG= 0x301D, 0x00      // IMAGE_ORIENTATION_
REG= 0x301E, 0x0028    // DATA_PEDESTAL_
REG= 0x3021, 0x00      // SOFTWARE_RESET_
REG= 0x3022, 0x00      // GROUPED_PARAMETER_HOLD_
REG= 0x3023, 0x00      // MASK_CORRUPTED_FRAMES_
REG= 0x3024, 0x00      // PIXEL_ORDER_
REG= 0x3026, 0xFFFF    // GPI_STATUS
REG= 0x3028, 0x0004    // ANALOGUE_GAIN_CODE_GLOBAL_
REG= 0x302A, 0x0004    // ANALOGUE_GAIN_CODE_GREENR_
REG= 0x302C, 0x0004    // ANALOGUE_GAIN_CODE_RED_
REG= 0x302E, 0x0004    // ANALOGUE_GAIN_CODE_BLUE_
REG= 0x3030, 0x0004    // ANALOGUE_GAIN_CODE_GREENB_
REG= 0x3032, 0x0100    // DIGITAL_GAIN_GREENR_
```

```
REG= 0x3034, 0x0100    // DIGITAL_GAIN_RED_

REG= 0x3036, 0x0100    // DIGITAL_GAIN_BLUE_

REG= 0x3038, 0x0100    // DIGITAL_GAIN_GREENB_

REG= 0x303A, 0x0A      // SMIA_VERSION_

REG= 0x303B, 0xB9      // FRAME_COUNT_

REG= 0x303C, 0x0000    // FRAME_STATUS

REG= 0x3040, 0x28C3    // READ_MODE

REG= 0x3044, 0x0590    // RESERVED_MFR_3044

REG= 0x3046, 0x0600    // FLASH

REG= 0x3048, 0x0008    // FLASH_COUNT

REG= 0x304A, 0x0020    // RESERVED_MFR_304A

REG= 0x304C, 0x0200    // RESERVED_MFR_304C

REG= 0x304E, 0x0000    // RESERVED_MFR_304E

REG= 0x3050, 0x0000    // RESERVED_MFR_3050

REG= 0x3052, 0x2174    // RESERVED_MFR_3052

REG= 0x3054, 0x0000    // RESERVED_MFR_3054

REG= 0x3056, 0x1020    // GREEN1_GAIN

REG= 0x3058, 0x1020    // BLUE_GAIN

REG= 0x305A, 0x1020    // RED_GAIN

REG= 0x305C, 0x1020    // GREEN2_GAIN

REG= 0x305E, 0x1020    // GLOBAL_GAIN

REG= 0x3060, 0x1500    // RESERVED_MFR_3060

REG= 0x3062, 0x0000    // RESERVED_MFR_3062

REG= 0x3064, 0x0905    // RESERVED_MFR_3064
```

```
REG= 0x3066, 0x0000    // RESERVED_MFR_3066
REG= 0x3068, 0x0000    // RESERVED_MFR_3068
REG= 0x306A, 0x0000    // DATAPATH_STATUS
REG= 0x306C, 0x1000    // RESERVED_MFR_306C
REG= 0x306E, 0x90B0    // DATAPATH_SELECT
REG= 0x3070, 0x0000    // TEST_PATTERN_MODE_
REG= 0x3072, 0x0000    // TEST_DATA_RED_
REG= 0x3074, 0x0000    // TEST_DATA_GREENR_
REG= 0x3076, 0x0000    // TEST_DATA_BLUE_
REG= 0x3078, 0x0000    // TEST_DATA_GREENB_
REG= 0x307A, 0x0000    // TEST_RAW_MODE
REG= 0x3080, 0x0000    // RESERVED_MFR_3080
REG= 0x30A0, 0x0001    // X_EVEN_INC_
REG= 0x30A2, 0x0003    // X_ODD_INC_
REG= 0x30A4, 0x0001    // Y_EVEN_INC_
REG= 0x30A6, 0x0003    // Y_ODD_INC_
REG= 0x30A8, 0x0004    // CALIB_GREEN1_ASC1
REG= 0x30AA, 0x0003    // CALIB_BLUE_ASC1
REG= 0x30AC, 0x0001    // CALIB_RED_ASC1
REG= 0x30AE, 0x0001    // CALIB_GREEN2_ASC1
REG= 0x30B0, 0x0003    // RESERVED_MFR_30B0
REG= 0x30B2, 0x8000    // RESERVED_MFR_30B2
REG= 0x30B4, 0x01FF    // RESERVED_MFR_30B4
REG= 0x30BC, 0x1000    // CALIB_GLOBAL
```

```
REG= 0x30C0, 0x0120    // CALIB_CONTROL
REG= 0x30C2, 0x0004    // CALIB_GREEN1
REG= 0x30C4, 0x0004    // CALIB_BLUE
REG= 0x30C6, 0x0001    // CALIB_RED
REG= 0x30C8, 0x0000    // CALIB_GREEN2
REG= 0x30CA, 0x8004    // RESERVED_MFR_30CA
REG= 0x30CC, 0x0000    // RESERVED_MFR_30CC
REG= 0x30CE, 0x0000    // RESERVED_MFR_30CE
REG= 0x30D0, 0x0FFF    // RESERVED_MFR_30D0
REG= 0x30D2, 0x0000    // RESERVED_MFR_30D2
REG= 0x30D4, 0x9080    // RESERVED_MFR_30D4
REG= 0x30D6, 0x0800    // RESERVED_MFR_30D6
REG= 0x30D8, 0x0000    // RESERVED_MFR_30D8
REG= 0x30DA, 0x0000    // RESERVED_MFR_30DA
REG= 0x30DC, 0x0000    // RESERVED_MFR_30DC
REG= 0x3130, 0x0F1F    // RESERVED_MFR_3130
REG= 0x3132, 0x0F1F    // RESERVED_MFR_3132
REG= 0x3134, 0x9F13    // RESERVED_MFR_3134
REG= 0x3136, 0x0404    // RESERVED_MFR_3136
REG= 0x3138, 0x4409    // RESERVED_MFR_3138
REG= 0x313A, 0x0000    // RESERVED_MFR_313A
REG= 0x313C, 0x0000    // RESERVED_MFR_313C
REG= 0x313E, 0x0000    // RESERVED_MFR_313E
REG= 0x315C, 0x0000    // RESERVED_MFR_315C
```

```
REG= 0x315E, 0x0000    // GLOBAL_SEQ_TRIGGER
REG= 0x3160, 0x0098    // GLOBAL_RST_END
REG= 0x3162, 0x00A8    // GLOBAL_SHUTTER_START
REG= 0x3164, 0x0000    // GLOBAL_SHUTTER_START2
REG= 0x3166, 0x00B8    // GLOBAL_READ_START
REG= 0x3168, 0x0000    // GLOBAL_READ_START2
REG= 0x316A, 0x0000    // RESERVED_MFR_316A
REG= 0x316C, 0x0429    // RESERVED_MFR_316C
REG= 0x316E, 0x0400    // RESERVED_MFR_316E
REG= 0x3170, 0x00E5    // RESERVED_MFR_3170
REG= 0x3172, 0x0501    // RESERVED_MFR_3172
REG= 0x3174, 0x8000    // RESERVED_MFR_3174
REG= 0x3176, 0x0000    // RESERVED_MFR_3176
REG= 0x3178, 0x0070    // RESERVED_MFR_3178
REG= 0x318A, 0x0006    // RESERVED_MFR_318A
REG= 0x318C, 0x0FFC    // RESERVED_MFR_318C
REG= 0x318E, 0x0FFD    // RESERVED_MFR_318E
REG= 0x3190, 0x0006    // RESERVED_MFR_3190
REG= 0x31A0, 0x0101    // DESCRIPTOR_0
REG= 0x31A2, 0x0201    // DESCRIPTOR_1
REG= 0x31A4, 0x0202    // DESCRIPTOR_2
REG= 0x31A6, 0x0301    // DESCRIPTOR_3
REG= 0x31A8, 0x0302    // DESCRIPTOR_4
REG= 0x31AA, 0x0304    // DESCRIPTOR_5
```

```
REG= 0x31AC, 0x0000    // DESCRIPTOR_6
REG= 0x31AE, 0x0304    // SERIAL_FORMAT
REG= 0x31B0, 0x0063    // FRAME_PREAMBLE
REG= 0x31B2, 0x0039    // LINE_PREAMBLE
REG= 0x31B4, 0x0D57    // MIPI_TIMING_0
REG= 0x31B6, 0x0B10    // MIPI_TIMING_1
REG= 0x31B8, 0x010D    // MIPI_TIMING_2
REG= 0x31BA, 0x050D    // MIPI_TIMING_3
REG= 0x31BC, 0x000B    // MIPI_TIMING_4
REG= 0x31BE, 0xC003    // RESERVED_MFR_31BE
REG= 0x31C0, 0x0000    // HISPI_TIMING
REG= 0x31C2, 0xFFFF    // RESERVED_MFR_31C2
REG= 0x31C4, 0xF555    // RESERVED_MFR_31C4
REG= 0x31C6, 0x8000    // HISPI_CONTROL_STATUS
REG= 0x31C8, 0x0000    // RESERVED_MFR_31C8
REG= 0x31CA, 0x0000    // RESERVED_MFR_31CA
REG= 0x31CC, 0x0000    // RESERVED_MFR_31CC
REG= 0x31CE, 0x0000    // RESERVED_MFR_31CE
REG= 0x31DA, 0x0000    // RESERVED_MFR_31DA
REG= 0x31DC, 0x0000    // RESERVED_MFR_31DC
REG= 0x31DE, 0x0000    // RESERVED_MFR_31DE
REG= 0x31E0, 0x0003    // RESERVED_MFR_31E0
REG= 0x31E2, 0x0000    // RESERVED_MFR_31E2
REG= 0x31E4, 0x0000    // RESERVED_MFR_31E4
```

```
REG= 0x31E8, 0x0000    // HORIZONTAL_CURSOR_POSITION_
REG= 0x31EA, 0x0000    // VERTICAL_CURSOR_POSITION_
REG= 0x31EC, 0x0000    // HORIZONTAL_CURSOR_WIDTH_
REG= 0x31EE, 0x0000    // VERTICAL_CURSOR_WIDTH_
REG= 0x31F2, 0x0000    // I2C_IDS_MIPI_DEFAULT
REG= 0x31F4, 0x0000    // RESERVED_MFR_31F4
REG= 0x31F6, 0x0000    // RESERVED_MFR_31F6
REG= 0x31F8, 0x0000    // RESERVED_MFR_31F8
REG= 0x31FA, 0x0000    // RESERVED_MFR_31FA
REG= 0x31FC, 0x3020    // I2C_IDS
REG= 0x31FE, 0x0032    // RESERVED_MFR_31FE
REG= 0x3600, 0x0850    // P_GR_P0Q0
REG= 0x3602, 0x0850    // P_GR_P0Q1
REG= 0x3604, 0x0850    // P_GR_P0Q2
REG= 0x3606, 0x0850    // P_GR_P0Q3
REG= 0x3608, 0x0850    // P_GR_P0Q4
REG= 0x360A, 0x0850    // P_RD_P0Q0
REG= 0x360C, 0x0850    // P_RD_P0Q1
REG= 0x360E, 0x0850    // P_RD_P0Q2
REG= 0x3610, 0x0850    // P_RD_P0Q3
REG= 0x3612, 0x0850    // P_RD_P0Q4
REG= 0x3614, 0x0850    // P_BL_P0Q0
REG= 0x3616, 0x0850    // P_BL_P0Q1
REG= 0x3618, 0x0850    // P_BL_P0Q2
```

```
REG= 0x361A, 0x0850    // P_BL_P0Q3
REG= 0x361C, 0x0850    // P_BL_P0Q4
REG= 0x361E, 0x0850    // P_GB_P0Q0
REG= 0x3620, 0x0850    // P_GB_P0Q1
REG= 0x3622, 0x0850    // P_GB_P0Q2
REG= 0x3624, 0x0850    // P_GB_P0Q3
REG= 0x3626, 0x0850    // P_GB_P0Q4
REG= 0x3640, 0xC20B    // P_GR_P1Q0
REG= 0x3642, 0xC20B    // P_GR_P1Q1
REG= 0x3644, 0xC20B    // P_GR_P1Q2
REG= 0x3646, 0xC20B    // P_GR_P1Q3
REG= 0x3648, 0xC20B    // P_GR_P1Q4
REG= 0x364A, 0xC20B    // P_RD_P1Q0
REG= 0x364C, 0xC20B    // P_RD_P1Q1
REG= 0x364E, 0xC20B    // P_RD_P1Q2
REG= 0x3650, 0x9C4A    // P_RD_P1Q3
REG= 0x3652, 0xC20B    // P_RD_P1Q4
REG= 0x3654, 0xC20B    // P_BL_P1Q0
REG= 0x3656, 0xC20B    // P_BL_P1Q1
REG= 0x3658, 0xC20B    // P_BL_P1Q2
REG= 0x365A, 0xC20B    // P_BL_P1Q3
REG= 0x365C, 0xC20B    // P_BL_P1Q4
REG= 0x365E, 0xC20B    // P_GB_P1Q0
REG= 0x3660, 0xC20B    // P_GB_P1Q1
```

```
REG= 0x3662, 0xC20B    // P_GB_P1Q2
REG= 0x3664, 0xC20B    // P_GB_P1Q3
REG= 0x3666, 0xC20B    // P_GB_P1Q4
REG= 0x3680, 0x6CAA    // P_GR_P2Q0
REG= 0x3682, 0x6CAA    // P_GR_P2Q1
REG= 0x3684, 0x6CAA    // P_GR_P2Q2
REG= 0x3686, 0x6CAA    // P_GR_P2Q3
REG= 0x3688, 0x6CAA    // P_GR_P2Q4
REG= 0x368A, 0x6CAA    // P_RD_P2Q0
REG= 0x368C, 0x6CAA    // P_RD_P2Q1
REG= 0x368E, 0x6CAA    // P_RD_P2Q2
REG= 0x3690, 0x6CAA    // P_RD_P2Q3
REG= 0x3692, 0x6CAA    // P_RD_P2Q4
REG= 0x3694, 0x6CAA    // P_BL_P2Q0
REG= 0x3696, 0x6CAA    // P_BL_P2Q1
REG= 0x3698, 0x6CAA    // P_BL_P2Q2
REG= 0x369A, 0x6CAA    // P_BL_P2Q3
REG= 0x369C, 0x6CAA    // P_BL_P2Q4
REG= 0x369E, 0x6CAA    // P_GB_P2Q0
REG= 0x36A0, 0x6CAA    // P_GB_P2Q1
REG= 0x36A2, 0x6CAA    // P_GB_P2Q2
REG= 0x36A4, 0x6CAA    // P_GB_P2Q3
REG= 0x36A6, 0x6CAA    // P_GB_P2Q4
REG= 0x36C0, 0x9A0A    // P_GR_P3Q0
```

```
REG= 0x36C2, 0x9A0A   // P_GR_P3Q1
REG= 0x36C4, 0x9A0A   // P_GR_P3Q2
REG= 0x36C6, 0x9A0A   // P_GR_P3Q3
REG= 0x36C8, 0x9A0A   // P_GR_P3Q4
REG= 0x36CA, 0x9A0A   // P_RD_P3Q0
REG= 0x36CC, 0x9A0A   // P_RD_P3Q1
REG= 0x36CE, 0x9A0A   // P_RD_P3Q2
REG= 0x36D0, 0x9A0A   // P_RD_P3Q3
REG= 0x36D2, 0x9A0A   // P_RD_P3Q4
REG= 0x36D4, 0x9A0A   // P_BL_P3Q0
REG= 0x36D6, 0x9A0A   // P_BL_P3Q1
REG= 0x36D8, 0x9A0A   // P_BL_P3Q2
REG= 0x36DA, 0x9A0A   // P_BL_P3Q3
REG= 0x36DC, 0x9A0A   // P_BL_P3Q4
REG= 0x36DE, 0x9A0A   // P_GB_P3Q0
REG= 0x36E0, 0x9A0A   // P_GB_P3Q1
REG= 0x36E2, 0x9A0A   // P_GB_P3Q2
REG= 0x36E4, 0x9A0A   // P_GB_P3Q3
REG= 0x36E6, 0x9A0A   // P_GB_P3Q4
REG= 0x3700, 0xD06C   // P_GR_P4Q0
REG= 0x3702, 0xEE6C   // P_GR_P4Q1
REG= 0x3704, 0xEE6C   // P_GR_P4Q2
REG= 0x3706, 0xEE6C   // P_GR_P4Q3
REG= 0x3708, 0xEE6C   // P_GR_P4Q4
```

```
REG= 0x370A, 0xEE6C    // P_RD_P4Q0
REG= 0x370C, 0xEE6C    // P_RD_P4Q1
REG= 0x370E, 0xEE6C    // P_RD_P4Q2
REG= 0x3710, 0xEE6C    // P_RD_P4Q3
REG= 0x3712, 0xEE6C    // P_RD_P4Q4
REG= 0x3714, 0xEE6C    // P_BL_P4Q0
REG= 0x3716, 0xEE6C    // P_BL_P4Q1
REG= 0x3718, 0xEE6C    // P_BL_P4Q2
REG= 0x371A, 0xEE6C    // P_BL_P4Q3
REG= 0x371C, 0xEE6C    // P_BL_P4Q4
REG= 0x371E, 0xEE6C    // P_GB_P4Q0
REG= 0x3720, 0xEE6C    // P_GB_P4Q1
REG= 0x3722, 0xEE6C    // P_GB_P4Q2
REG= 0x3724, 0xEE6C    // P_GB_P4Q3
REG= 0x3726, 0xEE6C    // P_GB_P4Q4
REG= 0x3780, 0x8000    // POLY_SC_ENABLE
REG= 0x3782, 0x07BC    // POLY_ORIGIN_C
REG= 0x3784, 0x0544    // POLY_ORIGIN_R
REG= 0x3800, 0x0000    // RESERVED_MFR_3800
REG= 0x3802, 0x0000    // RESERVED_MFR_3802
REG= 0x3804, 0x0000    // RESERVED_MFR_3804
REG= 0x3806, 0x0000    // RESERVED_MFR_3806
REG= 0x3808, 0x0000    // RESERVED_MFR_3808
REG= 0x380A, 0x0000    // RESERVED_MFR_380A
```

```
REG= 0x380C, 0x0000    // RESERVED_MFR_380C
REG= 0x380E, 0x0000    // RESERVED_MFR_380E
REG= 0x3810, 0x0000    // RESERVED_MFR_3810
REG= 0x3812, 0x0000    // RESERVED_MFR_3812
REG= 0x3814, 0x0000    // RESERVED_MFR_3814
REG= 0x3816, 0x0000    // RESERVED_MFR_3816
REG= 0x3818, 0x0000    // RESERVED_MFR_3818
REG= 0x381A, 0x0000    // RESERVED_MFR_381A
REG= 0x381C, 0x0000    // RESERVED_MFR_381C
REG= 0x381E, 0x0000    // RESERVED_MFR_381E
REG= 0x3820, 0x0000    // RESERVED_MFR_3820
REG= 0x3822, 0x0000    // RESERVED_MFR_3822
REG= 0x3824, 0x0000    // RESERVED_MFR_3824
REG= 0x3826, 0x0000    // RESERVED_MFR_3826
REG= 0x3828, 0x0000    // RESERVED_MFR_3828
REG= 0x382A, 0x0000    // RESERVED_MFR_382A
REG= 0x382C, 0x0000    // RESERVED_MFR_382C
REG= 0x382E, 0x0000    // RESERVED_MFR_382E
REG= 0x3830, 0x0000    // RESERVED_MFR_3830
REG= 0x3832, 0x0000    // RESERVED_MFR_3832
REG= 0x3834, 0x0000    // RESERVED_MFR_3834
REG= 0x3836, 0x0000    // RESERVED_MFR_3836
REG= 0x3838, 0x0000    // RESERVED_MFR_3838
REG= 0x383A, 0x0000    // RESERVED_MFR_383A
```

```
REG= 0x383C, 0x0000    // RESERVED_MFR_383C
REG= 0x383E, 0x0000    // RESERVED_MFR_383E
REG= 0x3840, 0x0000    // RESERVED_MFR_3840
REG= 0x3842, 0x0000    // RESERVED_MFR_3842
REG= 0x3844, 0x0000    // RESERVED_MFR_3844
REG= 0x3846, 0x0000    // RESERVED_MFR_3846
REG= 0x3848, 0x0000    // RESERVED_MFR_3848
REG= 0x384A, 0x0000    // RESERVED_MFR_384A
REG= 0x384C, 0x0000    // RESERVED_MFR_384C
REG= 0x384E, 0x0000    // RESERVED_MFR_384E
REG= 0x3850, 0x0000    // RESERVED_MFR_3850
REG= 0x3852, 0x0000    // RESERVED_MFR_3852
REG= 0x3854, 0x0000    // RESERVED_MFR_3854
REG= 0x3856, 0x0000    // RESERVED_MFR_3856
REG= 0x3858, 0x0000    // RESERVED_MFR_3858
REG= 0x385A, 0x0000    // RESERVED_MFR_385A
REG= 0x385C, 0x0000    // RESERVED_MFR_385C
REG= 0x385E, 0x0000    // RESERVED_MFR_385E
REG= 0x3860, 0x0000    // RESERVED_MFR_3860
REG= 0x3862, 0x0000    // RESERVED_MFR_3862
REG= 0x3864, 0x0000    // RESERVED_MFR_3864
REG= 0x3866, 0x0000    // RESERVED_MFR_3866
REG= 0x3868, 0x0000    // RESERVED_MFR_3868
REG= 0x386A, 0x0000    // RESERVED_MFR_386A
```

```
REG= 0x386C, 0x0000    // RESERVED_MFR_386C
REG= 0x386E, 0x0000    // RESERVED_MFR_386E
REG= 0x3870, 0x0000    // RESERVED_MFR_3870
REG= 0x3872, 0x0000    // RESERVED_MFR_3872
REG= 0x3874, 0x0000    // RESERVED_MFR_3874
REG= 0x3876, 0x0000    // RESERVED_MFR_3876
REG= 0x3878, 0x0000    // RESERVED_MFR_3878
REG= 0x387A, 0x0000    // RESERVED_MFR_387A
REG= 0x387C, 0x0000    // RESERVED_MFR_387C
REG= 0x387E, 0x0000    // RESERVED_MFR_387E
REG= 0x3880, 0x0000    // RESERVED_MFR_3880
REG= 0x3882, 0x0000    // RESERVED_MFR_3882
REG= 0x3884, 0x0000    // RESERVED_MFR_3884
REG= 0x3886, 0x0000    // RESERVED_MFR_3886
REG= 0x3888, 0x0000    // RESERVED_MFR_3888
REG= 0x388A, 0x0000    // RESERVED_MFR_388A
REG= 0x388C, 0x0000    // RESERVED_MFR_388C
REG= 0x388E, 0x0000    // RESERVED_MFR_388E
REG= 0x3890, 0x0000    // RESERVED_MFR_3890
REG= 0x3892, 0x0000    // RESERVED_MFR_3892
REG= 0x3894, 0x0000    // RESERVED_MFR_3894
REG= 0x3896, 0x0000    // RESERVED_MFR_3896
REG= 0x3898, 0x0000    // RESERVED_MFR_3898
REG= 0x389A, 0x0000    // RESERVED_MFR_389A
```

```
REG= 0x389C, 0x0000    // RESERVED_MFR_389C
REG= 0x389E, 0x0000    // RESERVED_MFR_389E
REG= 0x38A0, 0x0000    // RESERVED_MFR_38A0
REG= 0x38A2, 0x0000    // RESERVED_MFR_38A2
REG= 0x38A4, 0x0000    // RESERVED_MFR_38A4
REG= 0x38A6, 0x0000    // RESERVED_MFR_38A6
REG= 0x38A8, 0x0000    // RESERVED_MFR_38A8
REG= 0x38AA, 0x0000    // RESERVED_MFR_38AA
REG= 0x38AC, 0x0000    // RESERVED_MFR_38AC
REG= 0x38AE, 0x0000    // RESERVED_MFR_38AE
REG= 0x38B0, 0x0000    // RESERVED_MFR_38B0
REG= 0x38B2, 0x0000    // RESERVED_MFR_38B2
REG= 0x38B4, 0x0000    // RESERVED_MFR_38B4
REG= 0x38B6, 0x0000    // RESERVED_MFR_38B6
REG= 0x38B8, 0x0000    // RESERVED_MFR_38B8
REG= 0x38BA, 0x0000    // RESERVED_MFR_38BA
REG= 0x38BC, 0x0000    // RESERVED_MFR_38BC
REG= 0x38BE, 0x0000    // RESERVED_MFR_38BE
REG= 0x38C0, 0x0000    // RESERVED_MFR_38C0
REG= 0x38C2, 0x0000    // RESERVED_MFR_38C2
REG= 0x38C4, 0x0000    // RESERVED_MFR_38C4
REG= 0x38C6, 0x0000    // RESERVED_MFR_38C6
REG= 0x38C8, 0x0000    // RESERVED_MFR_38C8
REG= 0x38CA, 0x0000    // RESERVED_MFR_38CA
```

```
REG= 0x38CC, 0x0000    // RESERVED_MFR_38CC
REG= 0x38CE, 0x0000    // RESERVED_MFR_38CE
REG= 0x38D0, 0x0000    // RESERVED_MFR_38D0
REG= 0x38D2, 0x0000    // RESERVED_MFR_38D2
REG= 0x38D4, 0x0000    // RESERVED_MFR_38D4
REG= 0x38D6, 0x0000    // RESERVED_MFR_38D6
REG= 0x38D8, 0x0000    // RESERVED_MFR_38D8
REG= 0x38DA, 0x0000    // RESERVED_MFR_38DA
REG= 0x38DC, 0x0000    // RESERVED_MFR_38DC
REG= 0x38DE, 0x0000    // RESERVED_MFR_38DE
REG= 0x38E0, 0x0000    // RESERVED_MFR_38E0
REG= 0x38E2, 0x0000    // RESERVED_MFR_38E2
REG= 0x38E4, 0x0000    // RESERVED_MFR_38E4
REG= 0x38E6, 0x0000    // RESERVED_MFR_38E6
REG= 0x38E8, 0x0000    // RESERVED_MFR_38E8
REG= 0x38EA, 0x0000    // RESERVED_MFR_38EA
REG= 0x38EC, 0x0000    // RESERVED_MFR_38EC
REG= 0x38EE, 0x0000    // RESERVED_MFR_38EE
REG= 0x38F0, 0x0000    // RESERVED_MFR_38F0
REG= 0x38F2, 0x0000    // RESERVED_MFR_38F2
REG= 0x38F4, 0x0000    // RESERVED_MFR_38F4
REG= 0x38F6, 0x0000    // RESERVED_MFR_38F6
REG= 0x38F8, 0x0000    // RESERVED_MFR_38F8
REG= 0x38FA, 0x0000    // RESERVED_MFR_38FA
```

REG= 0x38FC, 0x0000 // RESERVED_MFR_38FC

REG= 0x38FE, 0x0000 // RESERVED_MFR_38FE

REG= 0x3900, 0x0000 // RESERVED_MFR_3900

REG= 0x3902, 0x0000 // RESERVED_MFR_3902

REG= 0x3904, 0x0000 // RESERVED_MFR_3904

REG= 0x3906, 0x0000 // RESERVED_MFR_3906

REG= 0x3908, 0x0000 // RESERVED_MFR_3908

REG= 0x390A, 0x0000 // RESERVED_MFR_390A

REG= 0x390C, 0x0000 // RESERVED_MFR_390C

REG= 0x390E, 0x0000 // RESERVED_MFR_390E

REG= 0x3910, 0x0000 // RESERVED_MFR_3910

REG= 0x3912, 0x0000 // RESERVED_MFR_3912

REG= 0x3914, 0x0000 // RESERVED_MFR_3914

REG= 0x3916, 0x0000 // RESERVED_MFR_3916

REG= 0x3918, 0x0000 // RESERVED_MFR_3918

REG= 0x391A, 0x0000 // RESERVED_MFR_391A

REG= 0x391C, 0x0000 // RESERVED_MFR_391C

REG= 0x391E, 0x0000 // RESERVED_MFR_391E

REG= 0x3920, 0x0000 // RESERVED_MFR_3920

REG= 0x3922, 0x0000 // RESERVED_MFR_3922

REG= 0x3924, 0x0000 // RESERVED_MFR_3924

REG= 0x3926, 0x0000 // RESERVED_MFR_3926

REG= 0x3928, 0x0000 // RESERVED_MFR_3928

REG= 0x392A, 0x0000 // RESERVED_MFR_392A

```
REG= 0x392C, 0x0000    // RESERVED_MFR_392C
REG= 0x392E, 0x0000    // RESERVED_MFR_392E
REG= 0x3930, 0x0000    // RESERVED_MFR_3930
REG= 0x3932, 0x0000    // RESERVED_MFR_3932
REG= 0x3934, 0x0000    // RESERVED_MFR_3934
REG= 0x3936, 0x0000    // RESERVED_MFR_3936
REG= 0x3938, 0x0000    // RESERVED_MFR_3938
REG= 0x393A, 0x0000    // RESERVED_MFR_393A
REG= 0x393C, 0x0000    // RESERVED_MFR_393C
REG= 0x393E, 0x0000    // RESERVED_MFR_393E
REG= 0x3940, 0x0000    // RESERVED_MFR_3940
REG= 0x3942, 0x0000    // RESERVED_MFR_3942
REG= 0x3944, 0x0000    // RESERVED_MFR_3944
REG= 0x3946, 0x0000    // RESERVED_MFR_3946
REG= 0x3948, 0x0000    // RESERVED_MFR_3948
REG= 0x394A, 0x0000    // RESERVED_MFR_394A
REG= 0x394C, 0x0000    // RESERVED_MFR_394C
REG= 0x394E, 0x0000    // RESERVED_MFR_394E
REG= 0x3950, 0x0000    // RESERVED_MFR_3950
REG= 0x3952, 0x0000    // RESERVED_MFR_3952
REG= 0x3954, 0x0000    // RESERVED_MFR_3954
REG= 0x3956, 0x0000    // RESERVED_MFR_3956
REG= 0x3958, 0x0000    // RESERVED_MFR_3958
REG= 0x395A, 0x0000    // RESERVED_MFR_395A
```

```
REG= 0x395C, 0x0000    // RESERVED_MFR_395C
REG= 0x395E, 0x0000    // RESERVED_MFR_395E
REG= 0x3960, 0x0000    // RESERVED_MFR_3960
REG= 0x3962, 0x0000    // RESERVED_MFR_3962
REG= 0x3964, 0x0000    // RESERVED_MFR_3964
REG= 0x3966, 0x0000    // RESERVED_MFR_3966
REG= 0x3968, 0x0000    // RESERVED_MFR_3968
REG= 0x396A, 0x0000    // RESERVED_MFR_396A
REG= 0x396C, 0x0000    // RESERVED_MFR_396C
REG= 0x396E, 0x0000    // RESERVED_MFR_396E
REG= 0x3970, 0x0000    // RESERVED_MFR_3970
REG= 0x3972, 0x0000    // RESERVED_MFR_3972
REG= 0x3974, 0x0000    // RESERVED_MFR_3974
REG= 0x3976, 0x0000    // RESERVED_MFR_3976
REG= 0x3978, 0x0000    // RESERVED_MFR_3978
REG= 0x397A, 0x0000    // RESERVED_MFR_397A
REG= 0x397C, 0x0000    // RESERVED_MFR_397C
REG= 0x397E, 0x0000    // RESERVED_MFR_397E
REG= 0x3980, 0x0000    // RESERVED_MFR_3980
REG= 0x3982, 0x0000    // RESERVED_MFR_3982
REG= 0x3984, 0x0000    // RESERVED_MFR_3984
REG= 0x3986, 0x0000    // RESERVED_MFR_3986
REG= 0x3988, 0x0000    // RESERVED_MFR_3988
REG= 0x398A, 0x0000    // RESERVED_MFR_398A
```

```
REG= 0x398C, 0x0000    // RESERVED_MFR_398C
REG= 0x398E, 0x0000    // RESERVED_MFR_398E
REG= 0x3990, 0x0000    // RESERVED_MFR_3990
REG= 0x3992, 0x0000    // RESERVED_MFR_3992
REG= 0x3994, 0x0000    // RESERVED_MFR_3994
REG= 0x3996, 0x0000    // RESERVED_MFR_3996
REG= 0x3998, 0x0000    // RESERVED_MFR_3998
REG= 0x399A, 0x0000    // RESERVED_MFR_399A
REG= 0x399C, 0x0000    // RESERVED_MFR_399C
REG= 0x399E, 0x0000    // RESERVED_MFR_399E
REG= 0x39A0, 0x0000    // RESERVED_MFR_39A0
REG= 0x39A2, 0x0000    // RESERVED_MFR_39A2
REG= 0x39A4, 0x0000    // RESERVED_MFR_39A4
REG= 0x39A6, 0x0000    // RESERVED_MFR_39A6
REG= 0x39A8, 0x0000    // RESERVED_MFR_39A8
REG= 0x39AA, 0x0000    // RESERVED_MFR_39AA
REG= 0x39AC, 0x0000    // RESERVED_MFR_39AC
REG= 0x39AE, 0x0000    // RESERVED_MFR_39AE
REG= 0x39B0, 0x0000    // RESERVED_MFR_39B0
REG= 0x39B2, 0x0000    // RESERVED_MFR_39B2
REG= 0x39B4, 0x0000    // RESERVED_MFR_39B4
REG= 0x39B6, 0x0000    // RESERVED_MFR_39B6
REG= 0x39B8, 0x0000    // RESERVED_MFR_39B8
REG= 0x39BA, 0x0000    // RESERVED_MFR_39BA
```

```
REG= 0x39BC, 0x0000    // RESERVED_MFR_39BC
REG= 0x39BE, 0x0000    // RESERVED_MFR_39BE
REG= 0x39C0, 0x0000    // RESERVED_MFR_39C0
REG= 0x39C2, 0x0000    // RESERVED_MFR_39C2
REG= 0x39C4, 0x0000    // RESERVED_MFR_39C4
REG= 0x39C6, 0x0000    // RESERVED_MFR_39C6
REG= 0x39C8, 0x0000    // RESERVED_MFR_39C8
REG= 0x39CA, 0x0000    // RESERVED_MFR_39CA
REG= 0x39CC, 0x0000    // RESERVED_MFR_39CC
REG= 0x39CE, 0x0000    // RESERVED_MFR_39CE
REG= 0x39D0, 0x0000    // RESERVED_MFR_39D0
REG= 0x39D2, 0x0000    // RESERVED_MFR_39D2
REG= 0x39D4, 0x0000    // RESERVED_MFR_39D4
REG= 0x39D6, 0x0000    // RESERVED_MFR_39D6
REG= 0x39D8, 0x0000    // RESERVED_MFR_39D8
REG= 0x39DA, 0x0000    // RESERVED_MFR_39DA
REG= 0x39DC, 0x0000    // RESERVED_MFR_39DC
REG= 0x39DE, 0x0000    // RESERVED_MFR_39DE
REG= 0x39E0, 0x0000    // RESERVED_MFR_39E0
REG= 0x39E2, 0x0000    // RESERVED_MFR_39E2
REG= 0x39E4, 0x0000    // RESERVED_MFR_39E4
REG= 0x39E6, 0x0000    // RESERVED_MFR_39E6
REG= 0x39E8, 0x0000    // RESERVED_MFR_39E8
REG= 0x39EA, 0x0000    // RESERVED_MFR_39EA
```

```
REG= 0x39EC, 0x0000    // RESERVED_MFR_39EC
REG= 0x39EE, 0x0000    // RESERVED_MFR_39EE
REG= 0x39F0, 0x0000    // RESERVED_MFR_39F0
REG= 0x39F2, 0x0000    // RESERVED_MFR_39F2
REG= 0x39F4, 0x0000    // RESERVED_MFR_39F4
REG= 0x39F6, 0x0000    // RESERVED_MFR_39F6
REG= 0x39F8, 0x0000    // RESERVED_MFR_39F8
REG= 0x39FA, 0x0000    // RESERVED_MFR_39FA
REG= 0x39FC, 0x0000    // RESERVED_MFR_39FC
REG= 0x39FE, 0x0000    // RESERVED_MFR_39FE
REG= 0x3E00, 0x0010    // RESERVED_MFR_3E00
REG= 0x3E02, 0xDE02    // RESERVED_MFR_3E02
REG= 0x3E04, 0x00FF    // RESERVED_MFR_3E04
REG= 0x3E06, 0x00FF    // RESERVED_MFR_3E06
REG= 0x3E08, 0xDC20    // RESERVED_MFR_3E08
REG= 0x3E0A, 0xDC06    // RESERVED_MFR_3E0A
REG= 0x3E0C, 0x3824    // RESERVED_MFR_3E0C
REG= 0x3E0E, 0x3425    // RESERVED_MFR_3E0E
REG= 0x3E10, 0x3622    // RESERVED_MFR_3E10
REG= 0x3E12, 0x0000    // RESERVED_MFR_3E12
REG= 0x3E14, 0xDA80    // RESERVED_MFR_3E14
REG= 0x3E16, 0x7C22    // RESERVED_MFR_3E16
REG= 0x3E18, 0x9C7E    // RESERVED_MFR_3E18
REG= 0x3E1A, 0x9980    // RESERVED_MFR_3E1A
```

```
REG= 0x3E1C, 0x9A7C    // RESERVED_MFR_3E1C
REG= 0x3E1E, 0x0000    // RESERVED_MFR_3E1E
REG= 0x3E20, 0xDC06    // RESERVED_MFR_3E20
REG= 0x3E22, 0x00FF    // RESERVED_MFR_3E22
REG= 0x3E24, 0xDC02    // RESERVED_MFR_3E24
REG= 0x3E26, 0xDC02    // RESERVED_MFR_3E26
REG= 0x3E28, 0xDC1E    // RESERVED_MFR_3E28
REG= 0x3E2A, 0xEE02    // RESERVED_MFR_3E2A
REG= 0x3E2C, 0x00FF    // RESERVED_MFR_3E2C
REG= 0x3E2E, 0x00FF    // RESERVED_MFR_3E2E
REG= 0x3E30, 0x00E8    // RESERVED_MFR_3E30
REG= 0x3E32, 0x52F0    // RESERVED_MFR_3E32
REG= 0x3E34, 0x0000    // RESERVED_MFR_3E34
REG= 0x3E36, 0x0000    // RESERVED_MFR_3E36
REG= 0x3E38, 0xDE04    // RESERVED_MFR_3E38
REG= 0x3E3A, 0xFF00    // RESERVED_MFR_3E3A
REG= 0x3E3C, 0x0000    // RESERVED_MFR_3E3C
REG= 0x3E3E, 0xDE02    // RESERVED_MFR_3E3E
REG= 0x3E40, 0xDC05    // RESERVED_MFR_3E40
REG= 0x3E42, 0x6E22    // RESERVED_MFR_3E42
REG= 0x3E44, 0xDC22    // RESERVED_MFR_3E44
REG= 0x3E46, 0xFF00    // RESERVED_MFR_3E46
REG= 0x3E48, 0xDC02    // RESERVED_MFR_3E48
REG= 0x3E4A, 0xDC02    // RESERVED_MFR_3E4A
```

```
REG= 0x3E4C, 0xDC1E    // RESERVED_MFR_3E4C
REG= 0x3E4E, 0xDC02    // RESERVED_MFR_3E4E
REG= 0x3E50, 0xDC1E    // RESERVED_MFR_3E50
REG= 0x3E52, 0xFF01    // RESERVED_MFR_3E52
REG= 0x3E54, 0x3222    // RESERVED_MFR_3E54
REG= 0x3E56, 0x00FF    // RESERVED_MFR_3E56
REG= 0x3E58, 0xDE04    // RESERVED_MFR_3E58
REG= 0x3E90, 0x5203    // RESERVED_MFR_3E90
REG= 0x3E92, 0x5005    // RESERVED_MFR_3E92
REG= 0x3E94, 0x4C06    // RESERVED_MFR_3E94
REG= 0x3E96, 0x4806    // RESERVED_MFR_3E96
REG= 0x3E98, 0x4607    // RESERVED_MFR_3E98
REG= 0x3E9A, 0x4A05    // RESERVED_MFR_3E9A
REG= 0x3E9C, 0x0000    // RESERVED_MFR_3E9C
REG= 0x3E9E, 0x4E04    // RESERVED_MFR_3E9E
REG= 0x3EA0, 0x0000    // RESERVED_MFR_3EA0
REG= 0x3EA2, 0x5200    // RESERVED_MFR_3EA2
REG= 0x3EB0, 0x0507    // RESERVED_MFR_3EB0
REG= 0x3EB2, 0x0608    // RESERVED_MFR_3EB2
REG= 0x3EB4, 0x97C7    // RESERVED_MFR_3EB4
REG= 0x3EB6, 0x97C6    // RESERVED_MFR_3EB6
REG= 0x3EB8, 0x0502    // RESERVED_MFR_3EB8
REG= 0x3ECC, 0x0FE4    // RESERVED_MFR_3ECC
REG= 0x3ECE, 0x1019    // RESERVED_MFR_3ECE
```

```
REG= 0x3ED0, 0x1B24    // RESERVED_MFR_3ED0
REG= 0x3ED2, 0xA660    // RESERVED_MFR_3ED2
REG= 0x3ED4, 0xF998    // RESERVED_MFR_3ED4
REG= 0x3ED6, 0x9789    // RESERVED_MFR_3ED6
REG= 0x3ED8, 0x5803    // RESERVED_MFR_3ED8
REG= 0x3EDA, 0xD9C3    // RESERVED_MFR_3EDA
REG= 0x3EDC, 0xD5E4    // RESERVED_MFR_3EDC
REG= 0x3EDE, 0xE41A    // RESERVED_MFR_3EDE
REG= 0x3EE0, 0xA43F    // RESERVED_MFR_3EE0
REG= 0x3EE2, 0xA4BF    // RESERVED_MFR_3EE2
REG= 0x3EE4, 0xE4E4    // RESERVED_MFR_3EE4
REG= 0x3EE6, 0x4540    // RESERVED_MFR_3EE6
REG= 0x3EE8, 0x0001    // RESERVED_MFR_3EE8
REG= 0x3EEA, 0x5500    // RESERVED_MFR_3EEA
REG= 0x3EEC, 0x1C21    // RESERVED_MFR_3EEC
REG= 0x3EEE, 0x1212    // RESERVED_MFR_3EEE
REG= 0x3EF0, 0x1212    // RESERVED_MFR_3EF0
```

APPENDIX C

Captured Image with Small Opening Lensless Camera

Captured image with small opening lensless camera

```
//  Captured 16:00:04 - Wednesday, June 11, 2014
//
//  Sensor info:
//          Width          = 1920
//          Height         = 1082
//          Image Format   = Bayer 12
//          Subformat      = GRBG
//          Sensor output  = 12 bits per pixel
//
//  .RAW file info:
//          stored as      = 16 bits unsigned short
//          valid data     = xxxxBA98 76543210
//          Endianess      = little
//
//  .BMP file info (24-bit windows bitmap file):
//          Width          = 1920
//          Height         = 1082
//
//  Application version info:
//          Application Name = C:\Aptina Imaging\DevWare.exe
//          Application Version = 4.5.23.39222
//          Application Type = 4.5.23_Release
```

```
//      Application Date = 05/05/2014
//
// Camera info:
//      Product ID = 0x100D Version = 0xC1
//      Product Name = Aptina Imaging DEMO2X
//      Firmware Version = D.2A
//      Transport Name = USB 2.0
//      Chip 0 Name = DEMO2X C1
//              CHIP_VERSION_REG = 0x00C1
//      Chip 1 Name = High Speed Serial Adapter
//              VERSION = 0x0131
//              CHIP_VERSION_REG = 0x00CD
//              DATESTAMPREG = 0x304A
//              TIMESTAMPREG = 0x1016
//      Sensor Name = A-10030
//      Sensor Part Number = MT9J003
//      Sensor Version = REV3
//      Sensor Filename = C:\Aptina Imaging\sensor_data\MT9J003-REV3.xsdat
//
// Sensor Fuse info:
//      FuseID: D652768C5758C679
//      Revision: 2
//      Silicon Option: --
//      Customer Revision: 0x32
```

```
//
// Windows OS info:
//      Display resolution = 1600x900 at 32bpp
//      OS Versioninfo = (6, 1, 7600, 2, , 0, 0)
//      Microsoft Windows Windows 7
//
//
// Processor info:
//      2095 MHz
//      GenuineIntel
//      Intel(R) Core(TM) i7-3612QM CPU @ 2.10GHz
//      49 percent of memory is in use.
//      Memory 8063 MB (total)
//      Memory 4048 MB (available)
//
// Display Devices:
//      Device 0: Intel(R) HD Graphics 4000
//
// USB Host Controllers:
//      Service: usbehci
//              Driver File: C:\Windows\system32\DRIVERS\usbehci.sys
//              File Version: 6.1.7601.18328
//              Device Desc 0:
PCI\VEN_8086&DEV_1E2D&SUBSYS_05641028&REV_04\3&11583659&0&D0
//              Device Desc 1:
```

PCI\VEN_8086&DEV_1E26&SUBSYS_05641028&REV_04\3&11583659&0&E8

//

// Camera Driver Info:

// C:\Windows\System32\Drivers\USB64W7.SYS (3.4.1.20) - VendorID: 0x20FB (Aptina Imaging)

[ColorPipe State]

STATE= Display Zoom Percent, 33

STATE= Master Clock, 71000000

STATE= Update Sensor FPS, 1

STATE= Allow Update Sensor FPS, 1

STATE= Filter, 0

STATE= X Offset, 0

STATE= Y Offset, 0

STATE= Auto Offset, 1

STATE= CFA Pattern, 1

STATE= Gb-B Swap, 0

STATE= RGBC BiWindow, 2

STATE= Monochrome, 0

STATE= Bayer Quadrant, 0

STATE= Byte Swap, 0

STATE= RedBlue Swap, 0

STATE= True Black Scale, 4096

STATE= True Black Level, 40

STATE= True Black Enable, 2

STATE= Auto Luma Range, 1

STATE= Luma Lo, 0

STATE= Luma Hi, 255

STATE= Unswizzle Mode, 3

STATE= Swap 12-bit LSBs, 0

STATE= Column Repeat, 0

STATE= Orientation, 0

STATE= Deinterlace Mode, 3

STATE= Descramble Mode, 1

STATE= Special Pixel Mode, 0

STATE= Active Area Crop, 0

STATE= Output Channel, 0

STATE= Output BwColor, 0

STATE= X Bin, 1

STATE= Y Bin, 1

STATE= DVS Split Screen, 0

STATE= Stereo Merge, 0

STATE= Stereo Shift X, 0

STATE= Stereo Shift Y, 0

STATE= Stereo Colwise, 0

STATE= sRGB Color Standard, 0

STATE= Color Correction, 1

STATE= Gamma Correction, 45

STATE= Black Correct, 5

STATE= Saturation, 10

STATE= Contrast, 25

STATE= Aperture Enable, 1

STATE= Aperture, 4

STATE= Black CCM Kill Enable, 0

STATE= Black CCM Kill A, 240

STATE= Black CCM Kill B, 160

STATE= Black CCM Kill C, 80

STATE= Green Balance Enable, 0

STATE= Green Balance Apos, 128

STATE= Green Balance Bpos, 10

STATE= Green Balance Aneg, -128

STATE= Green Balance Bneg, 10

STATE= Auto Exposure, 0

STATE= Auto Exposure Target, 50

STATE= Auto Exposure Stability, 6

STATE= Auto Exposure Speed, 30

STATE= Auto Exposure Minimum FPS, 30

STATE= Auto Exposure Flicker Filter, 0

STATE= Auto Exposure Soft Limit, 33

STATE= Auto Exposure Soft Gain Limit, 40

STATE= Auto Exposure Gain Limit, 317

STATE= Auto Exposure Minimum Global Gain, 10

STATE= Auto Exposure Global Gain Limit, 10

STATE= Auto Exposure Software Gain Limit, 10

STATE= Auto Exposure Freeze Gains, 1

STATE= Auto Exposure Fade Saturation, 1

STATE= Auto Exposure Fade Aperture, 1

STATE= Auto Exposure Fade Target, 1

STATE= Auto Exposure Inner Zone, 50

STATE= Auto Exposure Outer Zone, 50

STATE= HDR AE Mode, 0

STATE= Software Gain, 1000

STATE= Mechanical Shutter Same, 1

STATE= Mechanical Shutter Time, 33333

STATE= Mechanical Shutter Delay, 0

STATE= Trigger Width, 0

STATE= White Balance, 1

STATE= WB Speed, 30

STATE= WB Adjust Gains, 0

STATE= WB Manual Position, 0

STATE= WB Manual RedGreen, 89

STATE= WB Manual BlueGreen, 127

STATE= WB Interpolate Saturation, 1

STATE= WB Normalize Matrix, 1

STATE= AWB Weight Map Method, 2

STATE= AWB Weight Map X Scale, 128

STATE= AWB Weight Map Y Scale, 256

STATE= AWB Weight Map X Shift, 32

STATE= AWB Weight Map Y Shift, 8

STATE= AWB Weight Map X Center, 1014

STATE= AWB Weight Map Y Center, 1009

STATE= AWB Weight Map Angle Sin, 48

STATE= AWB Weight Map Angle Cos, 43

STATE= AWB Weight Map Luma Low, 4

STATE= AWB Weight Map Luma High, 251

STATE= Minimum Gain, 1000

STATE= Show Min Gain As 1, 0

STATE= Default Relative Red Gain, 1000

STATE= Default Relative Blue Gain, 1570

STATE= Relative Red Gain, 1000

STATE= Relative Blue Gain, 1570

STATE= Lens Correction Enable, 0

STATE= Lens Correction Falloff, 100

STATE= Lens Correction Falloff R, 100

STATE= Lens Correction Falloff G1, 100

STATE= Lens Correction Falloff G2, 100

STATE= Lens Correction Falloff B, 100

STATE= Lens Correction Overlay, 0

STATE= Lens Correction Center X, 1920

STATE= Lens Correction Center Y, 1374

STATE= Lens Correction Coeff Prec, 16

STATE= Lens Correction Lens Radius, 0

STATE= Lens Correction Luma Only, 0

STATE= Lens Center Red X, 1920

STATE= Lens Center Red Y, 1374

STATE= Lens Center Green1 X, 1920

STATE= Lens Center Green1 Y, 1374

STATE= Lens Center Green2 X, 1920

STATE= Lens Center Green2 Y, 1374

STATE= Lens Center Blue X, 1920

STATE= Lens Center Blue Y, 1374

STATE= Lens Center Overlay, 0

STATE= Lens Sim Sensor, 0

STATE= Lens Sim Sensor Rev, 0

STATE= Lens Sim Enable Pwq, 0

STATE= Lens Sim Enable Poly, 0

STATE= Noise Removal, 45

STATE= Noise Removal Level, 99

STATE= Noise Removal Depth, 2

STATE= Noise Removal K1, 2000

STATE= Noise Removal K2, 1800

STATE= Noise Removal K3, 1000

STATE= Noise Removal Edges, 1

STATE= Noise Removal Kernel, 0

STATE= Optical Black, 0

STATE= Optical Black Row Filter, 1

STATE= Optical Black 1 Row Start, 0

STATE= Optical Black 1 Row End, 0

STATE= Optical Black 2 Row Start, 0

STATE= Optical Black 2 Row End, 0

STATE= Optical Black 1 Column Start, 0

STATE= Optical Black 1 Column End, 0

STATE= Optical Black 2 Column Start, 0

STATE= Optical Black 2 Column End, 0

STATE= ALTM Enable, 0

STATE= Defect Enable, 1

STATE= Defect Max, 10000

STATE= Defect Auto Defect Correction, 0

STATE= Flash Lamp, 0

STATE= Still Global Reset, 0

STATE= Global Reset Bulb, 0

STATE= Continuous GRR, 0

STATE= Num Capture Frames, 1

STATE= Still Mode, 0

STATE= Still Hold, 1

STATE= Still Capture Average, 0

STATE= Still Capture Timeout, 5

STATE= Still HalfPress, 0

STATE= Delay before snap, 0

STATE= Save 24bpp BMP, 1

STATE= Save RAW, 1

STATE= Save TXT, 1

STATE= Save HEX, 0

STATE= Save ITX, 0

STATE= Save CCR, 0

STATE= Save DXR, 0

STATE= Save 48bpp COLOR TIFF, 0

STATE= Save JPEG, 0

STATE= Save RAW JPEG, 0

STATE= Save BMP Info, 0

STATE= JPEG Quality (1-100), 98

STATE= Save RAW PNG, 0

STATE= Save PNG, 0

STATE= Save DNG, 0

STATE= Save SS, 0

STATE= Save Selection Rectangle, 0

STATE= ICC Profile, 0

STATE= Video Screen Capture, 1

STATE= VidCap Play FPS, 15000

STATE= VidCap Auto Play FPS, 1

STATE= VidCap Format, 0

STATE= VidCap FileName Increment, 1

STATE= RAM Capture, 0

STATE= RAM Capture MB, 128

STATE= RAM Capture Cycle, 1

STATE= Preview Recording, 1

STATE= WB Xenon Red Gain, 1024

STATE= WB Xenon Blue Gain, 1024

STATE= WB Led Red Gain, 1024

STATE= WB Led Blue Gain, 1024

STATE= MAE Overlay, 0

STATE= Noise Image Type, 0

STATE= Noise Frames, 50

STATE= Noise Defects, 0

STATE= Strip FSP, 1

STATE= Check Thumbnail Table, 0

STATE= CRA Overlay, 0

STATE= Allow FAR Access, 1

STATE= Pure Raw, 0

STATE= Sensor Orientation, 0

STATE= Clarity6, 3

STATE= Clarity7, 1

STATE= Dynamic Range LSB, 0

STATE= Dynamic Range MSB, 31

STATE= AI Repack, 1

STATE= AR1820 REV12 GHR Workaround, 0

STATE= Deinterleave HDR, 3

STATE= HDR AE Histogram High Percent, 0.99

STATE= HDR AE Histogram High Target, 0.75

STATE= HDR AE Scene Brightness Offset, 8

STATE= HDR AE Max Percent Adjust, 0

STATE= HDR AE Luma Target, 40 80 200 600 1000 1400 1800 2200

STATE= AWB Incandescent, 1.886 -1.248 0.362 -0.128 1.329 -0.201 -0.082 -0.775 1.857

STATE= AWB Sun, 2.036 -1.055 0.019 -0.086 1.351 -0.265 0.007 -0.551 1.544

STATE= AWB Incandescent Gain, 0.896 1.987

STATE= AWB Sun Gain, 1.691 0.957

STATE= WB Custom, 1 0 0 0 1 0 0 0 1

STATE= WB Custom Xenon, 1 0 0 0 1 0 0 0 1

STATE= WB Custom Led, 1 0 0 0 1 0 0 0 1

STATE= AWB Weight Map, 0 3 8224 4096 1 273 4866 4368 34 8995 8739 12816 308 17204 12612 17168 546 21349 4915 13361 291 21333 8241 12832 3 8755 4610 4640 17 275 0 0

STATE= Optical Black Level Rect, 0 0 0 0

STATE= Active Area Rect, -1 -1 -1 -1

STATE= Lens Curve Red,
C04192013743F8000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000

STATE= Lens Curve Green1,
C04192013743F8000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000

STATE= Lens Curve Green2,
C04192013743F8000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000

STATE= Lens Curve Blue,
C04192013743F8000000000000000000000000000000000000000000000000000000000000

000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000

STATE= RGBC Sigma_S, 1

STATE= RGBC Sigma_I, 0.005

STATE= RGBC Smooth_Th, 0.1

STATE= ALTM Peak Percent, 0.99

STATE= ALTM Sharp S, 1

STATE= HW Gain, 1

STATE= HW Global Gain, 1

STATE= HW Red Gain, 1

STATE= HW Green1 Gain, 1

STATE= HW Green2 Gain, 1

STATE= HW Blue Gain, 1.5625

STATE= HW Exposure Time, 0.666644

STATE= VidCap File, C:\Users\Aleksandar\Documents\Aptina Imaging\video.avi

STATE= Clarity2, 1

STATE= Clarity3, 0.04

STATE= Clarity4, 0

STATE= Clarity5, 0.34

STATE= Chroma Filter Absolute Add Corners, 0

STATE= Clarity23, 1 7 0

STATE= Clarity24, 24aff 1249e

STATE= Clarity8, 0.028

STATE= Clarity9, 0.12

STATE= Clarity25, 2 2

STATE= Clarity11, 0

STATE= Clarity12, 0.117

STATE= Clarity10, 0.05

STATE= Clarity22, 0.025

STATE= Clarity13, 0.04

STATE= Clarity14, 0.2

STATE= Clarity15, 8

STATE= Black Balance, 0 0 0 0

STATE= Clarity18, 1

STATE= Clarity17, 2

STATE= Clarity20, 0.24 0.17

STATE= Clarity19, 0.1 0.1

STATE= Clarity21, 0.05

STATE= Clarity16, 0.01

STATE= Clarity26, 0.1

STATE= Clarity27, 1

STATE= Post Gain, 1

STATE= NIR Factor Red, 1

STATE= NIR Factor Green, 1

STATE= NIR Factor Blue, 1

STATE= NIR Coeff Red, 0

STATE= NIR Coeff Green, 0

STATE= NIR Coeff Blue, 0

STATE= NIR WB Custom, 1 0 0 0 1 0 0 0 1

STATE= NIR WB Gains, 1 1 1

STATE= NIR Global Gain, 1

[Raw Image Format]

IMAGE= 1920, 1082, BAYER-12

[Register State]

```
REG= 0x0000, 0x2C01    // CHIP_VERSION_REG
REG= 0x0002, 0x20      // REVISION_NUMBER
REG= 0x0003, 0x06      // MANUFACTURER_ID
REG= 0x0004, 0x0A      // SMIA_VERSION
REG= 0x0005, 0xE7      // FRAME_COUNT
REG= 0x0006, 0x00      // PIXEL_ORDER
REG= 0x0008, 0x0028    // DATA_PEDESTAL
REG= 0x0040, 0x01      // FRAME_FORMAT_MODEL_TYPE
REG= 0x0041, 0x12      // FRAME_FORMAT_MODEL_SUBTYPE
REG= 0x0042, 0x5780    // FRAME_FORMAT_DESCRIPTOR_0
REG= 0x0044, 0x1002    // FRAME_FORMAT_DESCRIPTOR_1
REG= 0x0046, 0x5438    // FRAME_FORMAT_DESCRIPTOR_2
REG= 0x0048, 0x0000    // FRAME_FORMAT_DESCRIPTOR_3
REG= 0x004A, 0x0000    // FRAME_FORMAT_DESCRIPTOR_4
REG= 0x004C, 0x0000    // FRAME_FORMAT_DESCRIPTOR_5
REG= 0x004E, 0x0000    // FRAME_FORMAT_DESCRIPTOR_6
REG= 0x0050, 0x0000    // FRAME_FORMAT_DESCRIPTOR_7
```

```
REG= 0x0052, 0x0000    // FRAME_FORMAT_DESCRIPTOR_8
REG= 0x0054, 0x0000    // FRAME_FORMAT_DESCRIPTOR_9
REG= 0x0056, 0x0000    // FRAME_FORMAT_DESCRIPTOR_10
REG= 0x0058, 0x0000    // FRAME_FORMAT_DESCRIPTOR_11
REG= 0x005A, 0x0000    // FRAME_FORMAT_DESCRIPTOR_12
REG= 0x005C, 0x0000    // FRAME_FORMAT_DESCRIPTOR_13
REG= 0x005E, 0x0000    // FRAME_FORMAT_DESCRIPTOR_14
REG= 0x0080, 0x0001    // ANALOGUE_GAIN_CAPABILITY
REG= 0x0084, 0x0008    // ANALOGUE_GAIN_CODE_MIN
REG= 0x0086, 0x00FF    // ANALOGUE_GAIN_CODE_MAX
REG= 0x0088, 0x0001    // ANALOGUE_GAIN_CODE_STEP
REG= 0x008A, 0x0000    // ANALOGUE_GAIN_TYPE
REG= 0x008C, 0x0001    // ANALOGUE_GAIN_M0
REG= 0x008E, 0x0000    // ANALOGUE_GAIN_C0
REG= 0x0090, 0x0000    // ANALOGUE_GAIN_M1
REG= 0x0092, 0x0008    // ANALOGUE_GAIN_C1
REG= 0x00C0, 0x01      // DATA_FORMAT_MODEL_TYPE
REG= 0x00C1, 0x05      // DATA_FORMAT_MODEL_SUBTYPE
REG= 0x00C2, 0x0A0A    // DATA_FORMAT_DESCRIPTOR_0
REG= 0x00C4, 0x0808    // DATA_FORMAT_DESCRIPTOR_1
REG= 0x00C6, 0x0A08    // DATA_FORMAT_DESCRIPTOR_2
REG= 0x00C8, 0x0C0C    // DATA_FORMAT_DESCRIPTOR_3
REG= 0x00CA, 0x0C08    // DATA_FORMAT_DESCRIPTOR_4
REG= 0x00CC, 0x0000    // DATA_FORMAT_DESCRIPTOR_5
```

```
REG= 0x00CE, 0x0000    // DATA_FORMAT_DESCRIPTOR_6
REG= 0x0100, 0x01      // MODE_SELECT
REG= 0x0101, 0x00      // IMAGE_ORIENTATION
REG= 0x0103, 0x00      // SOFTWARE_RESET
REG= 0x0104, 0x00      // GROUPED_PARAMETER_HOLD
REG= 0x0105, 0x00      // MASK_CORRUPTED_FRAMES
REG= 0x0110, 0x00      // CCP2_CHANNEL_IDENTIFIER
REG= 0x0111, 0x01      // CCP2_SIGNALLING_MODE
REG= 0x0112, 0x0C0C    // CCP_DATA_FORMAT
REG= 0x0120, 0x00      // GAIN_MODE
REG= 0x0200, 0x03F2    // FINE_INTEGRATION_TIME
REG= 0x0202, 0x5063    // COARSE_INTEGRATION_TIME
REG= 0x0204, 0x0008    // ANALOGUE_GAIN_CODE_GLOBAL
REG= 0x0206, 0x0008    // ANALOGUE_GAIN_CODE_GREENR
REG= 0x0208, 0x0008    // ANALOGUE_GAIN_CODE_RED
REG= 0x020A, 0x000C    // ANALOGUE_GAIN_CODE_BLUE
REG= 0x020C, 0x0008    // ANALOGUE_GAIN_CODE_GREENB
REG= 0x020E, 0x0100    // DIGITAL_GAIN_GREENR
REG= 0x0210, 0x0100    // DIGITAL_GAIN_RED
REG= 0x0212, 0x0100    // DIGITAL_GAIN_BLUE
REG= 0x0214, 0x0100    // DIGITAL_GAIN_GREENB
REG= 0x0300, 0x0003    // VT_PIX_CLK_DIV
REG= 0x0302, 0x0001    // VT_SYS_CLK_DIV
REG= 0x0304, 0x0003    // PRE_PLL_CLK_DIV
```

```
REG= 0x0306, 0x0030    // PLL_MULTIPLIER
REG= 0x0308, 0x000C    // OP_PIX_CLK_DIV
REG= 0x030A, 0x0001    // OP_SYS_CLK_DIV
REG= 0x0340, 0x048A    // FRAME_LENGTH_LINES
REG= 0x0342, 0x08FC    // LINE_LENGTH_PCK
REG= 0x0344, 0x0020    // X_ADDR_START
REG= 0x0346, 0x0128    // Y_ADDR_START
REG= 0x0348, 0x0F21    // X_ADDR_END
REG= 0x034A, 0x0995    // Y_ADDR_END
REG= 0x034C, 0x0780    // X_OUTPUT_SIZE
REG= 0x034E, 0x0438    // Y_OUTPUT_SIZE
REG= 0x0380, 0x0001    // X_EVEN_INC
REG= 0x0382, 0x0003    // X_ODD_INC
REG= 0x0384, 0x0001    // Y_EVEN_INC
REG= 0x0386, 0x0003    // Y_ODD_INC
REG= 0x0400, 0x0002    // SCALING_MODE
REG= 0x0402, 0x0000    // SPATIAL_SAMPLING
REG= 0x0404, 0x0010    // SCALE_M
REG= 0x0406, 0x0010    // SCALE_N
REG= 0x0500, 0x0001    // COMPRESSION_MODE
REG= 0x0600, 0x0000    // TEST_PATTERN_MODE
REG= 0x0602, 0x0000    // TEST_DATA_RED
REG= 0x0604, 0x0000    // TEST_DATA_GREENR
REG= 0x0606, 0x0000    // TEST_DATA_BLUE
```

```
REG= 0x0608, 0x0000     // TEST_DATA_GREENB
REG= 0x060A, 0x0000     // HORIZONTAL_CURSOR_WIDTH
REG= 0x060C, 0x0000     // HORIZONTAL_CURSOR_POSITION
REG= 0x060E, 0x0000     // VERTICAL_CURSOR_WIDTH
REG= 0x0610, 0x0000     // VERTICAL_CURSOR_POSITION
REG= 0x1000, 0x0001     // INTEGRATION_TIME_CAPABILITY
REG= 0x1004, 0x0000     // COARSE_INTEGRATION_TIME_MIN
REG= 0x1006, 0x0001     // COARSE_INTEGRATION_TIME_MAX_MARGIN
REG= 0x1008, 0x03F2     // FINE_INTEGRATION_TIME_MIN
REG= 0x100A, 0x027E     // FINE_INTEGRATION_TIME_MAX_MARGIN
REG= 0x1080, 0x0001     // DIGITAL_GAIN_CAPABILITY
REG= 0x1084, 0x0100     // DIGITAL_GAIN_MIN
REG= 0x1086, 0x0700     // DIGITAL_GAIN_MAX
REG= 0x1088, 0x0100     // DIGITAL_GAIN_STEP_SIZE
REG= 0x1100, 0x40000000         // MIN_EXT_CLK_FREQ_MHZ
REG= 0x1104, 0x42800000         // MAX_EXT_CLK_FREQ_MHZ
REG= 0x1108, 0x0001     // MIN_PRE_PLL_CLK_DIV
REG= 0x110A, 0x0040     // MAX_PRE_PLL_CLK_DIV
REG= 0x110C, 0x40800000         // MIN_PLL_IP_FREQ_MHZ
REG= 0x1110, 0x41C00000         // MAX_PLL_IP_FREQ_MHZ
REG= 0x1114, 0x0020     // MIN_PLL_MULTIPLIER
REG= 0x1116, 0x0180     // MAX_PLL_MULTIPLIER
REG= 0x1118, 0x43C00000         // MIN_PLL_OP_FREQ_MHZ
REG= 0x111C, 0x44400000         // MAX_PLL_OP_FREQ_MHZ
```

```
REG= 0x1120, 0x0001    // MIN_VT_SYS_CLK_DIV
REG= 0x1122, 0x0001    // MAX_VT_SYS_CLK_DIV
REG= 0x1124, 0x41C00000        // MIN_VT_SYS_CLK_FREQ_MHZ
REG= 0x1128, 0x44400000        // MAX_VT_SYS_CLK_FREQ_MHZ
REG= 0x112C, 0x4019999A        // MIN_VT_PIX_CLK_FREQ_MHZ
REG= 0x1130, 0x42C00000        // MAX_VT_PIX_CLK_FREQ_MHZ
REG= 0x1134, 0x0004    // MIN_VT_PIX_CLK_DIV
REG= 0x1136, 0x0006    // MAX_VT_PIX_CLK_DIV
REG= 0x1140, 0x0091    // MIN_FRAME_LENGTH_LINES
REG= 0x1142, 0xFFFF    // MAX_FRAME_LENGTH_LINES
REG= 0x1144, 0x0670    // MIN_LINE_LENGTH_PCK
REG= 0x1146, 0xFFFE    // MAX_LINE_LENGTH_PCK
REG= 0x1148, 0x046E    // MIN_LINE_BLANKING_PCK
REG= 0x114A, 0x008F    // MIN_FRAME_BLANKING_LINES
REG= 0x1160, 0x0001    // MIN_OP_SYS_CLK_DIV
REG= 0x1162, 0x0001    // MAX_OP_SYS_CLK_DIV
REG= 0x1164, 0x4199999A        // MIN_OP_SYS_CLK_FREQ_MHZ
REG= 0x1168, 0x44400000        // MAX_OP_SYS_CLK_FREQ_MHZ
REG= 0x116C, 0x0008    // MIN_OP_PIX_CLK_DIV
REG= 0x116E, 0x000C    // MAX_OP_PIX_CLK_DIV
REG= 0x1170, 0x4019999A        // MIN_OP_PIX_CLK_FREQ_MHZ
REG= 0x1174, 0x42C00000        // MAX_OP_PIX_CLK_FREQ_MHZ
REG= 0x1180, 0x0018    // X_ADDR_MIN
REG= 0x1182, 0x0000    // Y_ADDR_MIN
```

```
REG= 0x1184, 0x0F27    // X_ADDR_MAX
REG= 0x1186, 0x0ACB    // Y_ADDR_MAX
REG= 0x11C0, 0x0001    // MIN_EVEN_INC
REG= 0x11C2, 0x0001    // MAX_EVEN_INC
REG= 0x11C4, 0x0001    // MIN_ODD_INC
REG= 0x11C6, 0x0003    // MAX_ODD_INC
REG= 0x1200, 0x0002    // SCALING_CAPABILITY
REG= 0x1204, 0x0010    // SCALER_M_MIN
REG= 0x1206, 0x0080    // SCALER_M_MAX
REG= 0x1208, 0x0010    // SCALER_N_MIN
REG= 0x120A, 0x0010    // SCALER_N_MAX
REG= 0x1300, 0x0001    // COMPRESSION_CAPABILITY
REG= 0x1400, 0x0242    // MATRIX_ELEMENT_REDINRED
REG= 0x1402, 0xFF00    // MATRIX_ELEMENT_GREENINRED
REG= 0x1404, 0xFFBE    // MATRIX_ELEMENT_BLUEINRED
REG= 0x1406, 0xFFB4    // MATRIX_ELEMENT_REDINGREEN
REG= 0x1408, 0x0200    // MATRIX_ELEMENT_GREENINGREEN
REG= 0x140A, 0xFF4D    // MATRIX_ELEMENT_BLUEINGREEN
REG= 0x140C, 0xFFF1    // MATRIX_ELEMENT_REDINBLUE
REG= 0x140E, 0xFF34    // MATRIX_ELEMENT_GREENINBLUE
REG= 0x1410, 0x01DC    // MATRIX_ELEMENT_BLUEINBLUE
REG= 0x3000, 0x2C01    // MODEL_ID_
REG= 0x3002, 0x0128    // Y_ADDR_START_
REG= 0x3004, 0x0020    // X_ADDR_START_
```

```
REG= 0x3006, 0x0995    // Y_ADDR_END_
REG= 0x3008, 0x0F21    // X_ADDR_END_
REG= 0x300A, 0x048A    // FRAME_LENGTH_LINES_
REG= 0x300C, 0x08FC    // LINE_LENGTH_PCK_
REG= 0x3010, 0x009C    // FINE_CORRECTION
REG= 0x3012, 0x5063    // COARSE_INTEGRATION_TIME_
REG= 0x3014, 0x03F2    // FINE_INTEGRATION_TIME_
REG= 0x3016, 0x0121    // ROW_SPEED
REG= 0x3018, 0x0000    // EXTRA_DELAY
REG= 0x301A, 0x001C    // RESET_REGISTER
REG= 0x301C, 0x01      // MODE_SELECT_
REG= 0x301D, 0x00      // IMAGE_ORIENTATION_
REG= 0x301E, 0x0028    // DATA_PEDESTAL_
REG= 0x3021, 0x00      // SOFTWARE_RESET_
REG= 0x3022, 0x00      // GROUPED_PARAMETER_HOLD_
REG= 0x3023, 0x00      // MASK_CORRUPTED_FRAMES_
REG= 0x3024, 0x00      // PIXEL_ORDER_
REG= 0x3026, 0xFFFF    // GPI_STATUS
REG= 0x3028, 0x0008    // ANALOGUE_GAIN_CODE_GLOBAL_
REG= 0x302A, 0x0008    // ANALOGUE_GAIN_CODE_GREENR_
REG= 0x302C, 0x0008    // ANALOGUE_GAIN_CODE_RED_
REG= 0x302E, 0x000C    // ANALOGUE_GAIN_CODE_BLUE_
REG= 0x3030, 0x0008    // ANALOGUE_GAIN_CODE_GREENB_
REG= 0x3032, 0x0100    // DIGITAL_GAIN_GREENR_
```

```
REG= 0x3034, 0x0100    // DIGITAL_GAIN_RED_
REG= 0x3036, 0x0100    // DIGITAL_GAIN_BLUE_
REG= 0x3038, 0x0100    // DIGITAL_GAIN_GREENB_
REG= 0x303A, 0x0A      // SMIA_VERSION_
REG= 0x303B, 0xE7      // FRAME_COUNT_
REG= 0x303C, 0x0000    // FRAME_STATUS
REG= 0x3040, 0x28C3    // READ_MODE
REG= 0x3044, 0x0590    // RESERVED_MFR_3044
REG= 0x3046, 0x0600    // FLASH
REG= 0x3048, 0x0008    // FLASH_COUNT
REG= 0x304A, 0x0020    // RESERVED_MFR_304A
REG= 0x304C, 0x0200    // RESERVED_MFR_304C
REG= 0x304E, 0x0000    // RESERVED_MFR_304E
REG= 0x3050, 0x0000    // RESERVED_MFR_3050
REG= 0x3052, 0x2174    // RESERVED_MFR_3052
REG= 0x3054, 0x0000    // RESERVED_MFR_3054
REG= 0x3056, 0x1040    // GREEN1_GAIN
REG= 0x3058, 0x1064    // BLUE_GAIN
REG= 0x305A, 0x1040    // RED_GAIN
REG= 0x305C, 0x1040    // GREEN2_GAIN
REG= 0x305E, 0x1040    // GLOBAL_GAIN
REG= 0x3060, 0x1500    // RESERVED_MFR_3060
REG= 0x3062, 0x0000    // RESERVED_MFR_3062
REG= 0x3064, 0x0905    // RESERVED_MFR_3064
```

```
REG= 0x3066, 0x0000    // RESERVED_MFR_3066
REG= 0x3068, 0x0000    // RESERVED_MFR_3068
REG= 0x306A, 0x0000    // DATAPATH_STATUS
REG= 0x306C, 0x1000    // RESERVED_MFR_306C
REG= 0x306E, 0x90B0    // DATAPATH_SELECT
REG= 0x3070, 0x0000    // TEST_PATTERN_MODE_
REG= 0x3072, 0x0000    // TEST_DATA_RED_
REG= 0x3074, 0x0000    // TEST_DATA_GREENR_
REG= 0x3076, 0x0000    // TEST_DATA_BLUE_
REG= 0x3078, 0x0000    // TEST_DATA_GREENB_
REG= 0x307A, 0x0000    // TEST_RAW_MODE
REG= 0x3080, 0x0000    // RESERVED_MFR_3080
REG= 0x30A0, 0x0001    // X_EVEN_INC_
REG= 0x30A2, 0x0003    // X_ODD_INC_
REG= 0x30A4, 0x0001    // Y_EVEN_INC_
REG= 0x30A6, 0x0003    // Y_ODD_INC_
REG= 0x30A8, 0x0009    // CALIB_GREEN1_ASC1
REG= 0x30AA, 0x000C    // CALIB_BLUE_ASC1
REG= 0x30AC, 0x0001    // CALIB_RED_ASC1
REG= 0x30AE, 0x0002    // CALIB_GREEN2_ASC1
REG= 0x30B0, 0x0003    // RESERVED_MFR_30B0
REG= 0x30B2, 0x8000    // RESERVED_MFR_30B2
REG= 0x30B4, 0x01FF    // RESERVED_MFR_30B4
REG= 0x30BC, 0x1000    // CALIB_GLOBAL
```

```
REG= 0x30C0, 0x0120    // CALIB_CONTROL
REG= 0x30C2, 0x0009    // CALIB_GREEN1
REG= 0x30C4, 0x000C    // CALIB_BLUE
REG= 0x30C6, 0x0002    // CALIB_RED
REG= 0x30C8, 0x0002    // CALIB_GREEN2
REG= 0x30CA, 0x8004    // RESERVED_MFR_30CA
REG= 0x30CC, 0x0001    // RESERVED_MFR_30CC
REG= 0x30CE, 0x0FFE    // RESERVED_MFR_30CE
REG= 0x30D0, 0x0FFE    // RESERVED_MFR_30D0
REG= 0x30D2, 0x0000    // RESERVED_MFR_30D2
REG= 0x30D4, 0x9080    // RESERVED_MFR_30D4
REG= 0x30D6, 0x0800    // RESERVED_MFR_30D6
REG= 0x30D8, 0x0000    // RESERVED_MFR_30D8
REG= 0x30DA, 0x0000    // RESERVED_MFR_30DA
REG= 0x30DC, 0x0000    // RESERVED_MFR_30DC
REG= 0x3130, 0x0F1F    // RESERVED_MFR_3130
REG= 0x3132, 0x0F1F    // RESERVED_MFR_3132
REG= 0x3134, 0x9F13    // RESERVED_MFR_3134
REG= 0x3136, 0x0404    // RESERVED_MFR_3136
REG= 0x3138, 0x4409    // RESERVED_MFR_3138
REG= 0x313A, 0x0000    // RESERVED_MFR_313A
REG= 0x313C, 0x0000    // RESERVED_MFR_313C
REG= 0x313E, 0x0000    // RESERVED_MFR_313E
REG= 0x315C, 0x0000    // RESERVED_MFR_315C
```

```
REG= 0x315E, 0x0000    // GLOBAL_SEQ_TRIGGER
REG= 0x3160, 0x0098    // GLOBAL_RST_END
REG= 0x3162, 0x00A8    // GLOBAL_SHUTTER_START
REG= 0x3164, 0x0000    // GLOBAL_SHUTTER_START2
REG= 0x3166, 0x00B8    // GLOBAL_READ_START
REG= 0x3168, 0x0000    // GLOBAL_READ_START2
REG= 0x316A, 0x0000    // RESERVED_MFR_316A
REG= 0x316C, 0x0429    // RESERVED_MFR_316C
REG= 0x316E, 0x0400    // RESERVED_MFR_316E
REG= 0x3170, 0x00E5    // RESERVED_MFR_3170
REG= 0x3172, 0x0501    // RESERVED_MFR_3172
REG= 0x3174, 0x8000    // RESERVED_MFR_3174
REG= 0x3176, 0x0000    // RESERVED_MFR_3176
REG= 0x3178, 0x0070    // RESERVED_MFR_3178
REG= 0x318A, 0x0006    // RESERVED_MFR_318A
REG= 0x318C, 0x0FFE    // RESERVED_MFR_318C
REG= 0x318E, 0x0FFC    // RESERVED_MFR_318E
REG= 0x3190, 0x0005    // RESERVED_MFR_3190
REG= 0x31A0, 0x0101    // DESCRIPTOR_0
REG= 0x31A2, 0x0201    // DESCRIPTOR_1
REG= 0x31A4, 0x0202    // DESCRIPTOR_2
REG= 0x31A6, 0x0301    // DESCRIPTOR_3
REG= 0x31A8, 0x0302    // DESCRIPTOR_4
REG= 0x31AA, 0x0304    // DESCRIPTOR_5
```

```
REG= 0x31AC, 0x0000    // DESCRIPTOR_6
REG= 0x31AE, 0x0304    // SERIAL_FORMAT
REG= 0x31B0, 0x0063    // FRAME_PREAMBLE
REG= 0x31B2, 0x0039    // LINE_PREAMBLE
REG= 0x31B4, 0x0D57    // MIPI_TIMING_0
REG= 0x31B6, 0x0B10    // MIPI_TIMING_1
REG= 0x31B8, 0x010D    // MIPI_TIMING_2
REG= 0x31BA, 0x050D    // MIPI_TIMING_3
REG= 0x31BC, 0x000B    // MIPI_TIMING_4
REG= 0x31BE, 0xC003    // RESERVED_MFR_31BE
REG= 0x31C0, 0x0000    // HISPI_TIMING
REG= 0x31C2, 0xFFFF    // RESERVED_MFR_31C2
REG= 0x31C4, 0xF555    // RESERVED_MFR_31C4
REG= 0x31C6, 0x8000    // HISPI_CONTROL_STATUS
REG= 0x31C8, 0x0000    // RESERVED_MFR_31C8
REG= 0x31CA, 0x0000    // RESERVED_MFR_31CA
REG= 0x31CC, 0x0000    // RESERVED_MFR_31CC
REG= 0x31CE, 0x0000    // RESERVED_MFR_31CE
REG= 0x31DA, 0x0000    // RESERVED_MFR_31DA
REG= 0x31DC, 0x0000    // RESERVED_MFR_31DC
REG= 0x31DE, 0x0000    // RESERVED_MFR_31DE
REG= 0x31E0, 0x0003    // RESERVED_MFR_31E0
REG= 0x31E2, 0x0000    // RESERVED_MFR_31E2
REG= 0x31E4, 0x0000    // RESERVED_MFR_31E4
```

```
REG= 0x31E8, 0x0000    // HORIZONTAL_CURSOR_POSITION_
REG= 0x31EA, 0x0000    // VERTICAL_CURSOR_POSITION_
REG= 0x31EC, 0x0000    // HORIZONTAL_CURSOR_WIDTH_
REG= 0x31EE, 0x0000    // VERTICAL_CURSOR_WIDTH_
REG= 0x31F2, 0x0000    // I2C_IDS_MIPI_DEFAULT
REG= 0x31F4, 0x0000    // RESERVED_MFR_31F4
REG= 0x31F6, 0x0000    // RESERVED_MFR_31F6
REG= 0x31F8, 0x0000    // RESERVED_MFR_31F8
REG= 0x31FA, 0x0000    // RESERVED_MFR_31FA
REG= 0x31FC, 0x3020    // I2C_IDS
REG= 0x31FE, 0x0032    // RESERVED_MFR_31FE
REG= 0x3600, 0x0850    // P_GR_P0Q0
REG= 0x3602, 0x0850    // P_GR_P0Q1
REG= 0x3604, 0x0850    // P_GR_P0Q2
REG= 0x3606, 0x0850    // P_GR_P0Q3
REG= 0x3608, 0x0850    // P_GR_P0Q4
REG= 0x360A, 0x0850    // P_RD_P0Q0
REG= 0x360C, 0x0850    // P_RD_P0Q1
REG= 0x360E, 0x0850    // P_RD_P0Q2
REG= 0x3610, 0x0850    // P_RD_P0Q3
REG= 0x3612, 0x0850    // P_RD_P0Q4
REG= 0x3614, 0x0850    // P_BL_P0Q0
REG= 0x3616, 0x0850    // P_BL_P0Q1
REG= 0x3618, 0x0850    // P_BL_P0Q2
```

```
REG= 0x361A, 0x0850    // P_BL_P0Q3
REG= 0x361C, 0x0850    // P_BL_P0Q4
REG= 0x361E, 0x0850    // P_GB_P0Q0
REG= 0x3620, 0x0850    // P_GB_P0Q1
REG= 0x3622, 0x0850    // P_GB_P0Q2
REG= 0x3624, 0x0850    // P_GB_P0Q3
REG= 0x3626, 0x0850    // P_GB_P0Q4
REG= 0x3640, 0xC20B    // P_GR_P1Q0
REG= 0x3642, 0xC20B    // P_GR_P1Q1
REG= 0x3644, 0xC20B    // P_GR_P1Q2
REG= 0x3646, 0xC20B    // P_GR_P1Q3
REG= 0x3648, 0xC20B    // P_GR_P1Q4
REG= 0x364A, 0xC20B    // P_RD_P1Q0
REG= 0x364C, 0xC20B    // P_RD_P1Q1
REG= 0x364E, 0xC20B    // P_RD_P1Q2
REG= 0x3650, 0xC20B    // P_RD_P1Q3
REG= 0x3652, 0xC20B    // P_RD_P1Q4
REG= 0x3654, 0xC20B    // P_BL_P1Q0
REG= 0x3656, 0xC20B    // P_BL_P1Q1
REG= 0x3658, 0xC20B    // P_BL_P1Q2
REG= 0x365A, 0xC20B    // P_BL_P1Q3
REG= 0x365C, 0xC20B    // P_BL_P1Q4
REG= 0x365E, 0xC20B    // P_GB_P1Q0
REG= 0x3660, 0xC20B    // P_GB_P1Q1
```

```
REG= 0x3662, 0xC20B    // P_GB_P1Q2
REG= 0x3664, 0xC20B    // P_GB_P1Q3
REG= 0x3666, 0xC20B    // P_GB_P1Q4
REG= 0x3680, 0x6CAA    // P_GR_P2Q0
REG= 0x3682, 0x6CAA    // P_GR_P2Q1
REG= 0x3684, 0x6CAA    // P_GR_P2Q2
REG= 0x3686, 0x6CAA    // P_GR_P2Q3
REG= 0x3688, 0x6CAA    // P_GR_P2Q4
REG= 0x368A, 0x6CAA    // P_RD_P2Q0
REG= 0x368C, 0x6CAA    // P_RD_P2Q1
REG= 0x368E, 0xA6CA    // P_RD_P2Q2
REG= 0x3690, 0x6CAA    // P_RD_P2Q3
REG= 0x3692, 0x6CAA    // P_RD_P2Q4
REG= 0x3694, 0x6CAA    // P_BL_P2Q0
REG= 0x3696, 0x6CAA    // P_BL_P2Q1
REG= 0x3698, 0x6CAA    // P_BL_P2Q2
REG= 0x369A, 0x6CAA    // P_BL_P2Q3
REG= 0x369C, 0x6CAA    // P_BL_P2Q4
REG= 0x369E, 0x6CAA    // P_GB_P2Q0
REG= 0x36A0, 0x6CAA    // P_GB_P2Q1
REG= 0x36A2, 0x6CAA    // P_GB_P2Q2
REG= 0x36A4, 0x6CAA    // P_GB_P2Q3
REG= 0x36A6, 0x6CAA    // P_GB_P2Q4
REG= 0x36C0, 0x9A0A    // P_GR_P3Q0
```

```
REG= 0x36C2, 0x9A0A   // P_GR_P3Q1
REG= 0x36C4, 0x9A0A   // P_GR_P3Q2
REG= 0x36C6, 0x9A0A   // P_GR_P3Q3
REG= 0x36C8, 0x9A0A   // P_GR_P3Q4
REG= 0x36CA, 0x9A0A   // P_RD_P3Q0
REG= 0x36CC, 0x9A0A   // P_RD_P3Q1
REG= 0x36CE, 0x9A0A   // P_RD_P3Q2
REG= 0x36D0, 0x9A0A   // P_RD_P3Q3
REG= 0x36D2, 0x9A0A   // P_RD_P3Q4
REG= 0x36D4, 0x9A0A   // P_BL_P3Q0
REG= 0x36D6, 0x9A0A   // P_BL_P3Q1
REG= 0x36D8, 0x9A0A   // P_BL_P3Q2
REG= 0x36DA, 0x9A0A   // P_BL_P3Q3
REG= 0x36DC, 0x9A0A   // P_BL_P3Q4
REG= 0x36DE, 0x9A0A   // P_GB_P3Q0
REG= 0x36E0, 0x9A0A   // P_GB_P3Q1
REG= 0x36E2, 0x9A0A   // P_GB_P3Q2
REG= 0x36E4, 0x9A0A   // P_GB_P3Q3
REG= 0x36E6, 0x9A0A   // P_GB_P3Q4
REG= 0x3700, 0xEE6C   // P_GR_P4Q0
REG= 0x3702, 0xEE6C   // P_GR_P4Q1
REG= 0x3704, 0xEE6C   // P_GR_P4Q2
REG= 0x3706, 0xEE6C   // P_GR_P4Q3
REG= 0x3708, 0xEE6C   // P_GR_P4Q4
```

```
REG= 0x370A, 0xEE6C    // P_RD_P4Q0
REG= 0x370C, 0xEE6C    // P_RD_P4Q1
REG= 0x370E, 0xEE6C    // P_RD_P4Q2
REG= 0x3710, 0xEE6C    // P_RD_P4Q3
REG= 0x3712, 0xEE6C    // P_RD_P4Q4
REG= 0x3714, 0xEE6C    // P_BL_P4Q0
REG= 0x3716, 0xEE6C    // P_BL_P4Q1
REG= 0x3718, 0xEE6C    // P_BL_P4Q2
REG= 0x371A, 0xEE6C    // P_BL_P4Q3
REG= 0x371C, 0xEE6C    // P_BL_P4Q4
REG= 0x371E, 0xEE6C    // P_GB_P4Q0
REG= 0x3720, 0xEE6C    // P_GB_P4Q1
REG= 0x3722, 0xEE6C    // P_GB_P4Q2
REG= 0x3724, 0xEE6C    // P_GB_P4Q3
REG= 0x3726, 0xEE6C    // P_GB_P4Q4
REG= 0x3780, 0x8000    // POLY_SC_ENABLE
REG= 0x3782, 0x07BC    // POLY_ORIGIN_C
REG= 0x3784, 0x0544    // POLY_ORIGIN_R
REG= 0x3800, 0x0000    // RESERVED_MFR_3800
REG= 0x3802, 0x0000    // RESERVED_MFR_3802
REG= 0x3804, 0x0000    // RESERVED_MFR_3804
REG= 0x3806, 0x0000    // RESERVED_MFR_3806
REG= 0x3808, 0x0000    // RESERVED_MFR_3808
REG= 0x380A, 0x0000    // RESERVED_MFR_380A
```

```
REG= 0x380C, 0x0000    // RESERVED_MFR_380C
REG= 0x380E, 0x0000    // RESERVED_MFR_380E
REG= 0x3810, 0x0000    // RESERVED_MFR_3810
REG= 0x3812, 0x0000    // RESERVED_MFR_3812
REG= 0x3814, 0x0000    // RESERVED_MFR_3814
REG= 0x3816, 0x0000    // RESERVED_MFR_3816
REG= 0x3818, 0x0000    // RESERVED_MFR_3818
REG= 0x381A, 0x0000    // RESERVED_MFR_381A
REG= 0x381C, 0x0000    // RESERVED_MFR_381C
REG= 0x381E, 0x0000    // RESERVED_MFR_381E
REG= 0x3820, 0x0000    // RESERVED_MFR_3820
REG= 0x3822, 0x0000    // RESERVED_MFR_3822
REG= 0x3824, 0x0000    // RESERVED_MFR_3824
REG= 0x3826, 0x0000    // RESERVED_MFR_3826
REG= 0x3828, 0x0000    // RESERVED_MFR_3828
REG= 0x382A, 0x0000    // RESERVED_MFR_382A
REG= 0x382C, 0x0000    // RESERVED_MFR_382C
REG= 0x382E, 0x0000    // RESERVED_MFR_382E
REG= 0x3830, 0x0000    // RESERVED_MFR_3830
REG= 0x3832, 0x0000    // RESERVED_MFR_3832
REG= 0x3834, 0x0000    // RESERVED_MFR_3834
REG= 0x3836, 0x0000    // RESERVED_MFR_3836
REG= 0x3838, 0x0000    // RESERVED_MFR_3838
REG= 0x383A, 0x0000    // RESERVED_MFR_383A
```

```
REG= 0x383C, 0x0000    // RESERVED_MFR_383C
REG= 0x383E, 0x0000    // RESERVED_MFR_383E
REG= 0x3840, 0x0000    // RESERVED_MFR_3840
REG= 0x3842, 0x0000    // RESERVED_MFR_3842
REG= 0x3844, 0x0000    // RESERVED_MFR_3844
REG= 0x3846, 0x0000    // RESERVED_MFR_3846
REG= 0x3848, 0x0000    // RESERVED_MFR_3848
REG= 0x384A, 0x0000    // RESERVED_MFR_384A
REG= 0x384C, 0x0000    // RESERVED_MFR_384C
REG= 0x384E, 0x0000    // RESERVED_MFR_384E
REG= 0x3850, 0x0000    // RESERVED_MFR_3850
REG= 0x3852, 0x0000    // RESERVED_MFR_3852
REG= 0x3854, 0x0000    // RESERVED_MFR_3854
REG= 0x3856, 0x0000    // RESERVED_MFR_3856
REG= 0x3858, 0x0000    // RESERVED_MFR_3858
REG= 0x385A, 0x0000    // RESERVED_MFR_385A
REG= 0x385C, 0x0000    // RESERVED_MFR_385C
REG= 0x385E, 0x0000    // RESERVED_MFR_385E
REG= 0x3860, 0x0000    // RESERVED_MFR_3860
REG= 0x3862, 0x0000    // RESERVED_MFR_3862
REG= 0x3864, 0x0000    // RESERVED_MFR_3864
REG= 0x3866, 0x0000    // RESERVED_MFR_3866
REG= 0x3868, 0x0000    // RESERVED_MFR_3868
REG= 0x386A, 0x0000    // RESERVED_MFR_386A
```

```
REG= 0x386C, 0x0000    // RESERVED_MFR_386C
REG= 0x386E, 0x0000    // RESERVED_MFR_386E
REG= 0x3870, 0x0000    // RESERVED_MFR_3870
REG= 0x3872, 0x0000    // RESERVED_MFR_3872
REG= 0x3874, 0x0000    // RESERVED_MFR_3874
REG= 0x3876, 0x0000    // RESERVED_MFR_3876
REG= 0x3878, 0x0000    // RESERVED_MFR_3878
REG= 0x387A, 0x0000    // RESERVED_MFR_387A
REG= 0x387C, 0x0000    // RESERVED_MFR_387C
REG= 0x387E, 0x0000    // RESERVED_MFR_387E
REG= 0x3880, 0x0000    // RESERVED_MFR_3880
REG= 0x3882, 0x0000    // RESERVED_MFR_3882
REG= 0x3884, 0x0000    // RESERVED_MFR_3884
REG= 0x3886, 0x0000    // RESERVED_MFR_3886
REG= 0x3888, 0x0000    // RESERVED_MFR_3888
REG= 0x388A, 0x0000    // RESERVED_MFR_388A
REG= 0x388C, 0x0000    // RESERVED_MFR_388C
REG= 0x388E, 0x0000    // RESERVED_MFR_388E
REG= 0x3890, 0x0000    // RESERVED_MFR_3890
REG= 0x3892, 0x0000    // RESERVED_MFR_3892
REG= 0x3894, 0x0000    // RESERVED_MFR_3894
REG= 0x3896, 0x0000    // RESERVED_MFR_3896
REG= 0x3898, 0x0000    // RESERVED_MFR_3898
REG= 0x389A, 0x0000    // RESERVED_MFR_389A
```

```
REG= 0x389C, 0x0000    // RESERVED_MFR_389C
REG= 0x389E, 0x0000    // RESERVED_MFR_389E
REG= 0x38A0, 0x0000    // RESERVED_MFR_38A0
REG= 0x38A2, 0x0000    // RESERVED_MFR_38A2
REG= 0x38A4, 0x0000    // RESERVED_MFR_38A4
REG= 0x38A6, 0x0000    // RESERVED_MFR_38A6
REG= 0x38A8, 0x0000    // RESERVED_MFR_38A8
REG= 0x38AA, 0x0000    // RESERVED_MFR_38AA
REG= 0x38AC, 0x0000    // RESERVED_MFR_38AC
REG= 0x38AE, 0x0000    // RESERVED_MFR_38AE
REG= 0x38B0, 0x0000    // RESERVED_MFR_38B0
REG= 0x38B2, 0x0000    // RESERVED_MFR_38B2
REG= 0x38B4, 0x0000    // RESERVED_MFR_38B4
REG= 0x38B6, 0x0000    // RESERVED_MFR_38B6
REG= 0x38B8, 0x0000    // RESERVED_MFR_38B8
REG= 0x38BA, 0x0000    // RESERVED_MFR_38BA
REG= 0x38BC, 0x0000    // RESERVED_MFR_38BC
REG= 0x38BE, 0x0000    // RESERVED_MFR_38BE
REG= 0x38C0, 0x0000    // RESERVED_MFR_38C0
REG= 0x38C2, 0x0000    // RESERVED_MFR_38C2
REG= 0x38C4, 0x0000    // RESERVED_MFR_38C4
REG= 0x38C6, 0x0000    // RESERVED_MFR_38C6
REG= 0x38C8, 0x0000    // RESERVED_MFR_38C8
REG= 0x38CA, 0x0000    // RESERVED_MFR_38CA
```

```
REG= 0x38CC, 0x0000    // RESERVED_MFR_38CC

REG= 0x38CE, 0x0000    // RESERVED_MFR_38CE

REG= 0x38D0, 0x0000    // RESERVED_MFR_38D0

REG= 0x38D2, 0x0000    // RESERVED_MFR_38D2

REG= 0x38D4, 0x0000    // RESERVED_MFR_38D4

REG= 0x38D6, 0x0000    // RESERVED_MFR_38D6

REG= 0x38D8, 0x0000    // RESERVED_MFR_38D8

REG= 0x38DA, 0x0000    // RESERVED_MFR_38DA

REG= 0x38DC, 0x0000    // RESERVED_MFR_38DC

REG= 0x38DE, 0x0000    // RESERVED_MFR_38DE

REG= 0x38E0, 0x0000    // RESERVED_MFR_38E0

REG= 0x38E2, 0x0000    // RESERVED_MFR_38E2

REG= 0x38E4, 0x0000    // RESERVED_MFR_38E4

REG= 0x38E6, 0x0000    // RESERVED_MFR_38E6

REG= 0x38E8, 0x0000    // RESERVED_MFR_38E8

REG= 0x38EA, 0x0000    // RESERVED_MFR_38EA

REG= 0x38EC, 0x0000    // RESERVED_MFR_38EC

REG= 0x38EE, 0x0000    // RESERVED_MFR_38EE

REG= 0x38F0, 0x0000    // RESERVED_MFR_38F0

REG= 0x38F2, 0x0000    // RESERVED_MFR_38F2

REG= 0x38F4, 0x0000    // RESERVED_MFR_38F4

REG= 0x38F6, 0x0000    // RESERVED_MFR_38F6

REG= 0x38F8, 0x0000    // RESERVED_MFR_38F8

REG= 0x38FA, 0x0000    // RESERVED_MFR_38FA
```

```
REG= 0x38FC, 0x0000    // RESERVED_MFR_38FC
REG= 0x38FE, 0x0000    // RESERVED_MFR_38FE
REG= 0x3900, 0x0000    // RESERVED_MFR_3900
REG= 0x3902, 0x0000    // RESERVED_MFR_3902
REG= 0x3904, 0x0000    // RESERVED_MFR_3904
REG= 0x3906, 0x0000    // RESERVED_MFR_3906
REG= 0x3908, 0x0000    // RESERVED_MFR_3908
REG= 0x390A, 0x0000    // RESERVED_MFR_390A
REG= 0x390C, 0x0000    // RESERVED_MFR_390C
REG= 0x390E, 0x0000    // RESERVED_MFR_390E
REG= 0x3910, 0x0000    // RESERVED_MFR_3910
REG= 0x3912, 0x0000    // RESERVED_MFR_3912
REG= 0x3914, 0x0000    // RESERVED_MFR_3914
REG= 0x3916, 0x0000    // RESERVED_MFR_3916
REG= 0x3918, 0x0000    // RESERVED_MFR_3918
REG= 0x391A, 0x0000    // RESERVED_MFR_391A
REG= 0x391C, 0x0000    // RESERVED_MFR_391C
REG= 0x391E, 0x0000    // RESERVED_MFR_391E
REG= 0x3920, 0x0000    // RESERVED_MFR_3920
REG= 0x3922, 0x0000    // RESERVED_MFR_3922
REG= 0x3924, 0x0000    // RESERVED_MFR_3924
REG= 0x3926, 0x0000    // RESERVED_MFR_3926
REG= 0x3928, 0x0000    // RESERVED_MFR_3928
REG= 0x392A, 0x0000    // RESERVED_MFR_392A
```

```
REG= 0x392C, 0x0000    // RESERVED_MFR_392C
REG= 0x392E, 0x0000    // RESERVED_MFR_392E
REG= 0x3930, 0x0000    // RESERVED_MFR_3930
REG= 0x3932, 0x0000    // RESERVED_MFR_3932
REG= 0x3934, 0x0000    // RESERVED_MFR_3934
REG= 0x3936, 0x0000    // RESERVED_MFR_3936
REG= 0x3938, 0x0000    // RESERVED_MFR_3938
REG= 0x393A, 0x0000    // RESERVED_MFR_393A
REG= 0x393C, 0x0000    // RESERVED_MFR_393C
REG= 0x393E, 0x0000    // RESERVED_MFR_393E
REG= 0x3940, 0x0000    // RESERVED_MFR_3940
REG= 0x3942, 0x0000    // RESERVED_MFR_3942
REG= 0x3944, 0x0000    // RESERVED_MFR_3944
REG= 0x3946, 0x0000    // RESERVED_MFR_3946
REG= 0x3948, 0x0000    // RESERVED_MFR_3948
REG= 0x394A, 0x0000    // RESERVED_MFR_394A
REG= 0x394C, 0x0000    // RESERVED_MFR_394C
REG= 0x394E, 0x0000    // RESERVED_MFR_394E
REG= 0x3950, 0x0000    // RESERVED_MFR_3950
REG= 0x3952, 0x0000    // RESERVED_MFR_3952
REG= 0x3954, 0x0000    // RESERVED_MFR_3954
REG= 0x3956, 0x0000    // RESERVED_MFR_3956
REG= 0x3958, 0x0000    // RESERVED_MFR_3958
REG= 0x395A, 0x0000    // RESERVED_MFR_395A
```

```
REG= 0x395C, 0x0000    // RESERVED_MFR_395C
REG= 0x395E, 0x0000    // RESERVED_MFR_395E
REG= 0x3960, 0x0000    // RESERVED_MFR_3960
REG= 0x3962, 0x0000    // RESERVED_MFR_3962
REG= 0x3964, 0x0000    // RESERVED_MFR_3964
REG= 0x3966, 0x0000    // RESERVED_MFR_3966
REG= 0x3968, 0x0000    // RESERVED_MFR_3968
REG= 0x396A, 0x0000    // RESERVED_MFR_396A
REG= 0x396C, 0x0000    // RESERVED_MFR_396C
REG= 0x396E, 0x0000    // RESERVED_MFR_396E
REG= 0x3970, 0x0000    // RESERVED_MFR_3970
REG= 0x3972, 0x0000    // RESERVED_MFR_3972
REG= 0x3974, 0x0000    // RESERVED_MFR_3974
REG= 0x3976, 0x0000    // RESERVED_MFR_3976
REG= 0x3978, 0x0000    // RESERVED_MFR_3978
REG= 0x397A, 0x0000    // RESERVED_MFR_397A
REG= 0x397C, 0x0000    // RESERVED_MFR_397C
REG= 0x397E, 0x0000    // RESERVED_MFR_397E
REG= 0x3980, 0x0000    // RESERVED_MFR_3980
REG= 0x3982, 0x0000    // RESERVED_MFR_3982
REG= 0x3984, 0x0000    // RESERVED_MFR_3984
REG= 0x3986, 0x0000    // RESERVED_MFR_3986
REG= 0x3988, 0x0000    // RESERVED_MFR_3988
REG= 0x398A, 0x0000    // RESERVED_MFR_398A
```

REG= 0x398C, 0x0000 // RESERVED_MFR_398C

REG= 0x398E, 0x0000 // RESERVED_MFR_398E

REG= 0x3990, 0x0000 // RESERVED_MFR_3990

REG= 0x3992, 0x0000 // RESERVED_MFR_3992

REG= 0x3994, 0x0000 // RESERVED_MFR_3994

REG= 0x3996, 0x0000 // RESERVED_MFR_3996

REG= 0x3998, 0x0000 // RESERVED_MFR_3998

REG= 0x399A, 0x0000 // RESERVED_MFR_399A

REG= 0x399C, 0x0000 // RESERVED_MFR_399C

REG= 0x399E, 0x0000 // RESERVED_MFR_399E

REG= 0x39A0, 0x0000 // RESERVED_MFR_39A0

REG= 0x39A2, 0x0000 // RESERVED_MFR_39A2

REG= 0x39A4, 0x0000 // RESERVED_MFR_39A4

REG= 0x39A6, 0x0000 // RESERVED_MFR_39A6

REG= 0x39A8, 0x0000 // RESERVED_MFR_39A8

REG= 0x39AA, 0x0000 // RESERVED_MFR_39AA

REG= 0x39AC, 0x0000 // RESERVED_MFR_39AC

REG= 0x39AE, 0x0000 // RESERVED_MFR_39AE

REG= 0x39B0, 0x0000 // RESERVED_MFR_39B0

REG= 0x39B2, 0x0000 // RESERVED_MFR_39B2

REG= 0x39B4, 0x0000 // RESERVED_MFR_39B4

REG= 0x39B6, 0x0000 // RESERVED_MFR_39B6

REG= 0x39B8, 0x0000 // RESERVED_MFR_39B8

REG= 0x39BA, 0x0000 // RESERVED_MFR_39BA

```
REG= 0x39BC, 0x0000    // RESERVED_MFR_39BC
REG= 0x39BE, 0x0000    // RESERVED_MFR_39BE
REG= 0x39C0, 0x0000    // RESERVED_MFR_39C0
REG= 0x39C2, 0x0000    // RESERVED_MFR_39C2
REG= 0x39C4, 0x0000    // RESERVED_MFR_39C4
REG= 0x39C6, 0x0000    // RESERVED_MFR_39C6
REG= 0x39C8, 0x0000    // RESERVED_MFR_39C8
REG= 0x39CA, 0x0000    // RESERVED_MFR_39CA
REG= 0x39CC, 0x0000    // RESERVED_MFR_39CC
REG= 0x39CE, 0x0000    // RESERVED_MFR_39CE
REG= 0x39D0, 0x0000    // RESERVED_MFR_39D0
REG= 0x39D2, 0x0000    // RESERVED_MFR_39D2
REG= 0x39D4, 0x0000    // RESERVED_MFR_39D4
REG= 0x39D6, 0x0000    // RESERVED_MFR_39D6
REG= 0x39D8, 0x0000    // RESERVED_MFR_39D8
REG= 0x39DA, 0x0000    // RESERVED_MFR_39DA
REG= 0x39DC, 0x0000    // RESERVED_MFR_39DC
REG= 0x39DE, 0x0000    // RESERVED_MFR_39DE
REG= 0x39E0, 0x0000    // RESERVED_MFR_39E0
REG= 0x39E2, 0x0000    // RESERVED_MFR_39E2
REG= 0x39E4, 0x0000    // RESERVED_MFR_39E4
REG= 0x39E6, 0x0000    // RESERVED_MFR_39E6
REG= 0x39E8, 0x0000    // RESERVED_MFR_39E8
REG= 0x39EA, 0x0000    // RESERVED_MFR_39EA
```

```
REG= 0x39EC, 0x0000    // RESERVED_MFR_39EC
REG= 0x39EE, 0x0000    // RESERVED_MFR_39EE
REG= 0x39F0, 0x0000    // RESERVED_MFR_39F0
REG= 0x39F2, 0x0000    // RESERVED_MFR_39F2
REG= 0x39F4, 0x0000    // RESERVED_MFR_39F4
REG= 0x39F6, 0x0000    // RESERVED_MFR_39F6
REG= 0x39F8, 0x0000    // RESERVED_MFR_39F8
REG= 0x39FA, 0x0000    // RESERVED_MFR_39FA
REG= 0x39FC, 0x0000    // RESERVED_MFR_39FC
REG= 0x39FE, 0x0000    // RESERVED_MFR_39FE
REG= 0x3E00, 0x0010    // RESERVED_MFR_3E00
REG= 0x3E02, 0xDE02    // RESERVED_MFR_3E02
REG= 0x3E04, 0x00FF    // RESERVED_MFR_3E04
REG= 0x3E06, 0x00FF    // RESERVED_MFR_3E06
REG= 0x3E08, 0xDC20    // RESERVED_MFR_3E08
REG= 0x3E0A, 0xDC06    // RESERVED_MFR_3E0A
REG= 0x3E0C, 0x3824    // RESERVED_MFR_3E0C
REG= 0x3E0E, 0x3425    // RESERVED_MFR_3E0E
REG= 0x3E10, 0x3622    // RESERVED_MFR_3E10
REG= 0x3E12, 0x0000    // RESERVED_MFR_3E12
REG= 0x3E14, 0xDA80    // RESERVED_MFR_3E14
REG= 0x3E16, 0x7C22    // RESERVED_MFR_3E16
REG= 0x3E18, 0x9C7E    // RESERVED_MFR_3E18
REG= 0x3E1A, 0x9980    // RESERVED_MFR_3E1A
```

```
REG= 0x3E1C, 0x9A7C    // RESERVED_MFR_3E1C
REG= 0x3E1E, 0x0000    // RESERVED_MFR_3E1E
REG= 0x3E20, 0xDC06    // RESERVED_MFR_3E20
REG= 0x3E22, 0x00FF    // RESERVED_MFR_3E22
REG= 0x3E24, 0xDC02    // RESERVED_MFR_3E24
REG= 0x3E26, 0xDC02    // RESERVED_MFR_3E26
REG= 0x3E28, 0xDC1E    // RESERVED_MFR_3E28
REG= 0x3E2A, 0xEE02    // RESERVED_MFR_3E2A
REG= 0x3E2C, 0x00FF    // RESERVED_MFR_3E2C
REG= 0x3E2E, 0x00FF    // RESERVED_MFR_3E2E
REG= 0x3E30, 0x00E8    // RESERVED_MFR_3E30
REG= 0x3E32, 0x52F0    // RESERVED_MFR_3E32
REG= 0x3E34, 0x0000    // RESERVED_MFR_3E34
REG= 0x3E36, 0x0000    // RESERVED_MFR_3E36
REG= 0x3E38, 0xDE04    // RESERVED_MFR_3E38
REG= 0x3E3A, 0xFF00    // RESERVED_MFR_3E3A
REG= 0x3E3C, 0x0000    // RESERVED_MFR_3E3C
REG= 0x3E3E, 0xDE02    // RESERVED_MFR_3E3E
REG= 0x3E40, 0xDC05    // RESERVED_MFR_3E40
REG= 0x3E42, 0x6E22    // RESERVED_MFR_3E42
REG= 0x3E44, 0xDC22    // RESERVED_MFR_3E44
REG= 0x3E46, 0xFF00    // RESERVED_MFR_3E46
REG= 0x3E48, 0xDC02    // RESERVED_MFR_3E48
REG= 0x3E4A, 0xDC02    // RESERVED_MFR_3E4A
```

```
REG= 0x3E4C, 0xDC1E    // RESERVED_MFR_3E4C
REG= 0x3E4E, 0xDC02    // RESERVED_MFR_3E4E
REG= 0x3E50, 0xDC1E    // RESERVED_MFR_3E50
REG= 0x3E52, 0xFF01    // RESERVED_MFR_3E52
REG= 0x3E54, 0x3222    // RESERVED_MFR_3E54
REG= 0x3E56, 0x00FF    // RESERVED_MFR_3E56
REG= 0x3E58, 0xDE04    // RESERVED_MFR_3E58
REG= 0x3E90, 0x5203    // RESERVED_MFR_3E90
REG= 0x3E92, 0x5005    // RESERVED_MFR_3E92
REG= 0x3E94, 0x4C06    // RESERVED_MFR_3E94
REG= 0x3E96, 0x4806    // RESERVED_MFR_3E96
REG= 0x3E98, 0x4607    // RESERVED_MFR_3E98
REG= 0x3E9A, 0x4A05    // RESERVED_MFR_3E9A
REG= 0x3E9C, 0x0000    // RESERVED_MFR_3E9C
REG= 0x3E9E, 0x4E04    // RESERVED_MFR_3E9E
REG= 0x3EA0, 0x0000    // RESERVED_MFR_3EA0
REG= 0x3EA2, 0x5200    // RESERVED_MFR_3EA2
REG= 0x3EB0, 0x0507    // RESERVED_MFR_3EB0
REG= 0x3EB2, 0x0608    // RESERVED_MFR_3EB2
REG= 0x3EB4, 0x97C7    // RESERVED_MFR_3EB4
REG= 0x3EB6, 0x97C6    // RESERVED_MFR_3EB6
REG= 0x3EB8, 0x0502    // RESERVED_MFR_3EB8
REG= 0x3ECC, 0x0FE4    // RESERVED_MFR_3ECC
REG= 0x3ECE, 0x1019    // RESERVED_MFR_3ECE
```

```
REG= 0x3ED0, 0x1B24    // RESERVED_MFR_3ED0
REG= 0x3ED2, 0xA660    // RESERVED_MFR_3ED2
REG= 0x3ED4, 0xF998    // RESERVED_MFR_3ED4
REG= 0x3ED6, 0x9789    // RESERVED_MFR_3ED6
REG= 0x3ED8, 0x5803    // RESERVED_MFR_3ED8
REG= 0x3EDA, 0xD9C3    // RESERVED_MFR_3EDA
REG= 0x3EDC, 0xD5E4    // RESERVED_MFR_3EDC
REG= 0x3EDE, 0xE41A    // RESERVED_MFR_3EDE
REG= 0x3EE0, 0xA43F    // RESERVED_MFR_3EE0
REG= 0x3EE2, 0xA4BF    // RESERVED_MFR_3EE2
REG= 0x3EE4, 0xE4E4    // RESERVED_MFR_3EE4
REG= 0x3EE6, 0x4540    // RESERVED_MFR_3EE6
REG= 0x3EE8, 0x0001    // RESERVED_MFR_3EE8
REG= 0x3EEA, 0x5500    // RESERVED_MFR_3EEA
REG= 0x3EEC, 0x1C21    // RESERVED_MFR_3EEC
REG= 0x3EEE, 0x1212    // RESERVED_MFR_3EEE
REG= 0x3EF0, 0x1212    // RESERVED_MFR_3EF0
```