1. General Description:

The Faraya series of vision chips are a series of flexible resolution vision chips designed for a broad set of visual sensing applications. Three chips are available in this series, the Faraya256plus having a resolution of 256x256 pixels, the Faraya64plus having a resolution of 64x64 pixels, and the Faraya64mkii also having a resolution of 64x64 pixels but without the on-chip ADC and linear regulator. The Faraya64mkii is an updated version of the original Faraya64, which includes fixes such as the PIO12B interface and a modified focal plane circuit allowing usage at lower voltages. The general top-level features of chips in this series are as follows:

* Log-response mode pixel circuits
* Row amplifier circuits
* 2D switched capacitor array, one for each pixel, allowing binning by 2, 4, or 8 pixels, independently in each direction, including support for offset downsampling, as well as smoothing in each direction using switched capacitor techniques
* Direct addressing of pixels by row and column
* On-chip 8-bit flash ADC, with internal reference voltage generators (Faraya256plus and Faraya64plus only)
* On-chip linear voltage regulator (external bypass capacitor recommended) for analog circuitry (Faraya64plus only)
* PIO12B interface
2. General Specifications:

| Raw focal plane resolution | Faraya256plus: 256x256 (65536)  
|                           | Faraya64plus: 64x64 (4096)     
|                           | Faraya64mkii: 64x64 (4096)     |
| Drawn die size            | Faraya256plus: 11.1mm x 6.5mm  
|                           | Faraya64plus: 4.6mm x 3.2mm    
|                           | Faraya64mkii: 4.6mm x 2.9mm    |
| Focal plane size          | Faraya256plus: 4.4mm x 4.4mm   
|                           | Faraya64plus: 1.1mm x 1.1mm    
|                           | Faraya64mkii: 1.1mm x 1.1mm    |
| Pixel pitch               | 17.1 microns                   
| Pixel circuitry           | Continuous time log response   |
| Pixel binning             | 2D switched capacitor circuit  
|                           | Allows both simple binning and  
|                           | smoothing                       |
| Row amplification         | Four-level switched capacitor  
|                           | network                        
|                           | May be bypassed to allow raw   
|                           | pixels to be read out. Row     
|                           | amplification circuits located 
|                           | between focal plane array and  
|                           | switched capacitor array       |
| Interface                 | PIO12B                         |
| Process                   | ON-Semi 0.5u C5N 3M2P          |
| Power supply voltage      | VDD=5.0V (within C5N process   
|                           | spec)                           
|                           | VDD=6.5V (not guaranteed)      |

3. Chip layout:

The following pages show the layout of the Faraya256plus, Faraya64plus, and Faraya64mkii chips. The focal plane array and switched capacitor array are clearly visible as the blank squares in the layouts. This is because the individual circuits were too small to be properly rendered by the CAD tools.
Faraya64plus vision chip
4. List of pads / external signals:

Below is a list of all pads and external signals which is valid for all Faraya series chips. The pitch between pads is 90 microns. All pads having the same name are electrically connected on the chip. Most users will need to use only the subset of signals that are "black". Advanced and optional signals are in "blue" text.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VDD</td>
<td>Power</td>
<td>Main power to chip- provides power to most digital circuitry, non-critical analog circuitry, and to on-chip linear voltage regulator.</td>
</tr>
<tr>
<td>VDDI</td>
<td>Power, PIO12B</td>
<td>Provides power to PIO12B circuit for when providing output onto IO0...IO7 lines. Note that VDDI powers only the IO0...IO7 lines.</td>
</tr>
<tr>
<td>VDDA</td>
<td>Power, Linear Regulator</td>
<td>Analog power- generated by on-chip linear voltage regulator and powers critical analog circuits. Shorted to VDD on power-up.</td>
</tr>
<tr>
<td>GND</td>
<td>Power</td>
<td>Ground/substrate</td>
</tr>
<tr>
<td>IO0...IO7</td>
<td>PIO12B</td>
<td>8-bit parallel bidirectional command and data interface</td>
</tr>
<tr>
<td>CMD0...CMD11</td>
<td>PIO12B</td>
<td>Command lines when operating PIO12B in extended mode</td>
</tr>
<tr>
<td>WnR</td>
<td>PIO12B</td>
<td>Write / Not-Read PIO12B. Determines direction of signals IO0...IO7, internally pulled down (read)</td>
</tr>
<tr>
<td>CS1</td>
<td>PIO12B</td>
<td>Primary chip select line, internally pulled down. Active high</td>
</tr>
<tr>
<td>CS2, CS3</td>
<td>PIO12B</td>
<td>Secondary chip select lines, internally pulled up to VDD. Active high.</td>
</tr>
<tr>
<td>ANALOG0</td>
<td>Analog output</td>
<td>Primary analog output of pixel values</td>
</tr>
<tr>
<td>ANALOG1</td>
<td>Analog output</td>
<td>Second analog output of pixel values (Faraya256plus only)</td>
</tr>
<tr>
<td>OEPU</td>
<td>PIO12B</td>
<td>Output Enable PullUp- forces IO0...IO7 to be output e.g. forces OUTENABLE high. Internally pulled down. Note that OEPU1 and OEPU2 on the Faraya64plus and Faraya64mkii chips are connected together internally to form the OEPU signal.</td>
</tr>
<tr>
<td>OUTENABLE</td>
<td>PIO12B</td>
<td>Internally generated signal- &quot;1&quot; means IO0...IO7 lines and ANALOG1 are connected, &quot;0&quot; means disconnected. May be overrode but recommend using OEPU instead. Note: Digital high &quot;1&quot; is VDD.</td>
</tr>
<tr>
<td>LDCPU</td>
<td>PIO12B</td>
<td>LoaDCommand PullUp- forces a loadcommand operation e.g. forces LOADCMD high. Internally pulled down. Note that LDCPU1 and LDCPU2 on the Faraya64plus and Faraya64mkii chips are connected together internally to form the LDCPU signal.</td>
</tr>
<tr>
<td>LOADCMD</td>
<td>PIO12B</td>
<td>Internally generated signal- pulses to a &quot;1&quot; when a command is written to the chip. May be overrode but recommend using LDCPU instead. Note: Digital high &quot;1&quot; is VDD.</td>
</tr>
<tr>
<td>ENABLECMD611</td>
<td>PIO12B</td>
<td>Allows IO0...IO5 lines to be sent to CMD6...CMD11 lines. Internally pulled up to VDD.</td>
</tr>
<tr>
<td>--------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------</td>
</tr>
<tr>
<td>SHORTYUKPHI</td>
<td>ADC control</td>
<td>Can be used to operate the flash ADC directly, bypassing the PIO12B for faster operation. Set to &quot;1&quot; to reset ADC, release to &quot;0&quot; to begin conversion. ADCOP register must be configured properly for operation. Default is &quot;0&quot;. (Not included on Faraya64mkii)</td>
</tr>
<tr>
<td>PRSUPPLY</td>
<td>Bias (vaulted)</td>
<td>Supply for photoreceptor circuits when in log response mode.</td>
</tr>
<tr>
<td>NBIAS</td>
<td>Bias (vaulted)</td>
<td>Supply for column readout circuits</td>
</tr>
<tr>
<td>ANALOGOUTBIAS</td>
<td>Bias (vaulted)</td>
<td>Bias for output analog amplifier</td>
</tr>
<tr>
<td>YUKNBIAS</td>
<td>Bias</td>
<td>Bias for Yukawa latch in on-chip flash ADC. (Not included on Faraya64mkii)</td>
</tr>
<tr>
<td>VREF</td>
<td>Bias (vaulted)</td>
<td>Reference voltage for row amplifiers</td>
</tr>
<tr>
<td>RESB</td>
<td>Bias for ADC</td>
<td>Lower (bottom) reference voltage for flash ADC. (Not included on Faraya64mkii)</td>
</tr>
<tr>
<td>REST</td>
<td>Bias for ADC</td>
<td>Upper (top) reference voltage for flash ADC. (Not included on Faraya64mkii)</td>
</tr>
<tr>
<td>BIASSWITCH</td>
<td>Bias Control</td>
<td>State of bias vault- &quot;1&quot; is open, &quot;0&quot; is closed. Default is &quot;0&quot;. Generally never used as an input to the chip, but may be monitored to verify chip operation. Note: potential at &quot;1&quot; is VDD, not VDD.</td>
</tr>
<tr>
<td>CONNECTVDDA</td>
<td>Bias Control (vaulted)</td>
<td>Determines whether bias circuits are powered- &quot;0&quot; is powered, &quot;1&quot; is not powered. Default is &quot;1&quot;. Note: potential at &quot;1&quot; is VDD, not VDDI.</td>
</tr>
<tr>
<td>ESVR</td>
<td>Linear Regulator Control</td>
<td>Enable Short VReg. &quot;1&quot; means linear regulator disabled and VDD shorted to VDDA. &quot;0&quot; means linear regulator turned on. Default is &quot;1&quot; at VDD via internal pull-ups. May be controlled by setting LINREG registers. (Not included on Faraya64mkii)</td>
</tr>
</tbody>
</table>
5. Connecting a Faraya chip to power and to a processor:

5.1 Connecting a Faraya256plus or Faraya64plus chip

The figure below shows a minimal connection between a Faraya vision chip and a processor operating it. This configuration is for when using the PIO12B interface in standard mode. Refer to documentation on the PIO12B interface for more details. If the processor and the Faraya vision chips will be operated from the same power supply, then VDD and VDDI may be shorted together. DO NOT let VDDI exceed VDD in potential. The VDD and VDDA power rails each should have at least one bypass capacitor between the power rail and ground. The optimal capacitances have not yet been determined, but it is suggested that at least a larger (1uF or above) capacitor be used for each of these. Additional capacitors and capacitance will not hurt the chip and will likely help. Note that on-chip power up VDD and VDDA are internally shorted together by default, and are disconnected as part of setting up the linear regulator (see Section 8.2). Optionally one may externally short VDDA and VDD together.

![Faraya chip diagram](image)

5.2 Connecting a Faraya64mkii chip

A Faraya64mkii may be similarly connected, except that 1) there is no on-chip linear voltage regulator therefore an external VDDA signal must be supplied, 2) there is no on-chip ADC therefore the analog signal must be digitized, either by an external ADC or by an ADC included in the processor, and 3) only IO0...IO5 and IO7 lines are needed, and these are on-directional from the processor to the Faraya64mkii chip.
6. Commands:

All Centeye vision chips using the PIO12B interface use 12-bit commands for all operations. These commands may be used to program on-chip biases, configure the chip for a particular operation, operate the chip itself, or select a specific pixel by row and column for readout. Each command may take the form of a three-bit header and a nine-bit argument, or a six-bit header and a six-bit argument. In both cases the headers form the most significant bits. Refer to the documentation of the vision chip for information on the specific command codes and their usage.

The vision chip may be connected in one of two ways: The default method is "standard PIO12B mode", in which the 12-bit commands are written to the vision chip in two six-bit sets. This may sound wasteful, however once the upper six-bits are written to the PIO12B interface, they are stored until overwritten again. In general there are only one or two commands that dominate 98% of the commands written, thus the upper six bit portion of the 12-bit command may be left untouched, thus requiring that only the lower six bits be sent. Therefore in practice the penalty is minimal.

The other method of connecting the vision chip is in "extended PIO12B mode", in which the 12-bit commands are written directly to the vision chip through separate signals CMD0...CMD11. Refer to Centeye documentation on the PIO12B interface for more details.

7. Chip Block Diagram:

7.1 Overview

The figure below shows a high level block diagram of the chip. The PIO12B module serves as the entry and exit point for communications with the chip. The PIO12B module sends 12-bit commands to registers that handle configuration, chip operation, and pixel selection for readout. Additional 12-bit commands control on-chip bias generators, some vaulted and some non-vaulted. Vaulted biases have a key circuit (the "vault") which must be enabled to allow them to be changed- this is a safety feature to prevent against setting of biases during operation.
The focal plane comprises a 256x256 or 64x64 array of pixel circuits, which are discussed further below. Row select signals are generated on the left of the focal plane and are used to read out rows. There is one one row select line for each row of pixels. Column readout lines carry the output values of the selected pixels to a row of amplifier circuits at bottom of the focal plane. There is one column readout line and one row amplifier for each column of pixels.

The amplifier circuits amplify the column signals (or optionally pass them through unchanged) and send them to a capacitor array which holds a capacitor array the same size as the pixel array, so that one pixel is linked with one capacitor. When the load row signal for a row is pulsed, the row of capacitors are charged to the potential generated by the row of amplifiers. Thus the row of capacitors "stores" the row of pixel values. This processed may be repeated for every row so that the capacitor array stores a single frame of pixels from the focal plane. Switches located between adjacent capacitors in the capacitor array may then be operated to short blocks of capacitors together so they contain the same charge, which effectively generates a single potential based on the average of all pixel values within the block. This will be discussed further below.

When the "read row" signal for a row is high, the potentials stored on the selected row of capacitors is sent to the bottom of the capacitor array, where a column select circuit selects one of these values as the pixel output.

An analog buffer circuit buffers the selected signal and sends it to the 8-bit flash ADC for conversion. If the "OUTENABLE" signal is high (e.g. WNR=0 and all CSs=1) then the buffered signal is also sent off chip to the ANALOG1 pad.

An 8-bit flash ADC converts the buffered pixel value into an 8-bit value that is sent to the PIO12B interface. When the chip is selected to be read, this 8-bit value becomes the digital output on lines IO0...IO7. Biases RESB and REST are the lower and upper reference voltages for the ADC. The signal SHORTYUKPHI allows the ADC to be operated directly from off-chip, bypassing the PIO12B module for speed. The ADC may also be operated through the PIO12B module.

A linear voltage regulator generates the VDDA power rail from VDD, and is also configured through the PIO12B.
7.2 Pixel Circuit:

The pixel circuit is shown below. The light sensing element is N-well photodiode D1 (formed between an N-well and the P-substrate, the latter tied to ground). As light strikes the photodiode D1, it sinks current to the substrate (to ground) proportional to the amount of light striking it. This current is typically in the picoamp or nanoamp range. This small current flows through transistors M1 and M2. Due to the small current, these transistors are operating in the subthreshold region. Since these transistors are diode connected, the voltage drop across M1 and M2 will be a logarithmic function of the current flowing through them, and thus a logarithmic function of the intensity. When light changes by a factor of $e=2.7$, the voltage drop across M1 and M2 will change by about 100mV to 150mV. A large range of light intensities may thus be compressed within a manageable voltage swing. The bias PRSUPPLY may be adjusted to shift the level of the output. The voltage generated by the pixel circuit is located at the node with the circle (•).

When the row select signal is high, and when the "switched VDD" supply is on (by setting VDDSWITCH=0), the voltage is read out to the column line through transistors M3 and M4. The row select signal for row i may be turned high by setting ROWSEL to i and setting the READPIX signal high. The column line goes to transistor MC, which with the bias NBIAS1 and transistor M3 forms a source follower to output the pixel value. Note that there is only one MC transistor for each column line.
7.3 Row Amplifier Circuit:

The row amplifier array contains one amplifier circuit for each column line. This amplifier circuit allows the column signals to be processed by one of two ways.

If SELAMP is high, then the column signal is amplified using an adjustable gain switched capacitor circuit. Essentially the switched capacitor circuit compares the column voltage with bias VREF. The polarity of the inputs may be as shown in the figure or swapped. The amplifier may be configured for one of four gains.

If SELUNITY is high, then the column signal is sent out without amplification or shifting. The row of pixel values are then sent to the capacitor array.

![Diagram of Row Amplifier Circuit](image-url)
7.4 Capacitor Array:

The capacitor array is a 2D array of capacitor circuits, with one capacitor circuit for each pixel. Thus the Faraya256plus chip contains a 256×256 array of capacitor circuits, while the Faraya64plus and Faraya64mkii chips contain a 64×64 array of capacitor circuits.

The schematic for a capacitor circuit is shown below. When the "load row" row signal is pulsed high, capacitor C1 is set to a potential equal to the column input potential. One row of the capacitor array is "loaded" in this manner at a time. In this manner, when the "row select" value is set to row i, the pixel array will output row i to an array of row amplifier circuits as shown above. The amplified (if SELAMP=1) or passed-through (if SELUNITY=1) row signals are provided to the column inputs of the switched capacitor array. When the "load row" signal pulses high, the corresponding row of C1 capacitors is loaded with the set of potentials generated by the array of amplifiers. To set the load row signal for row i, first set ROWSEL to i (if it is not already set to this value) and pulse signal LOADCAP high. This process can be repeated for every row to load the capacitor array with the contents of the focal plane array, with or without row amplification as desired. When loading the capacitor array as described, the variables VSW and HSW should be all zero.

Once the capacitor array is loaded with the potentials, the vsw and hsw switches may be turned on to short out adjacent capacitors. This implements "binning" where the voltages according to a neighborhood block of pixels may be averaged together. The hsw switch transistors (e.g. M5) are operated by the HSW command and are as follows: Bit 0 of HSW closes the switches between columns 0 and 1, between columns 8 and 9, between columns 16 and 17, and so on. Bit 1 of HSW closes the switches between columns 1 and 2, between columns 9 and 10, and so on. Bits 2 through 7 of HSW close the switches between the other columns in the same manner.
Similarly, the vsw switch transistors (e.g. M4) close the switches between rows of capacitor circuits in a similar manner, except the order is as follows: Bit 1 of VSW closes the vsw switches between rows 0 and 1, rows 16 and 17, and so on. Bit 2 of VSW closes the switches between rows 1 and 2, rows 17 and 18, and so on. Bit 7 of VSW closes the vsw switches between rows 6 and 7, rows 14 and 15, and so on. Finally, Bit 0 of VSW closes the vsw switches between rows 7 and 8, rows 15 and 16, and so on.

"Super pixels" may be implemented by binning by closing the appropriate VSW and HSW signals:
* To bin horizontally by a factor of 2, set HSW to binary 01010101 (bit 7 is on the left = "0"), e.g. by performing Send(HSW+b'01010101')
* To bin horizontally by a factor of 4, set HSW to binary 01110111
* To bin horizontally by a factor of 8, set HSW to binary 01111111
* To bin vertically by a factor of 8, set VSW to binary 11111110 (bit 7 is on the left = "1")
* To bin vertically by a factor of 4, set VSW to binary 11101110
* To bin vertically by a factor of 2, set VSW to binary 10101010
* To not bin in either direction, leave VSW or HSW as appropriate at 0000000 (all zeros).

Note that the capacitor network can be binned horizontally and vertically independently by different amounts simply by setting HSW and VSW as needed. Once binning is complete, it is advised to set VSW and HSW back to all zero.

For example, to bin the pixels down by a factor of 4 in each direction, the following steps may be used:

* Set HSW and VSW to 00000000 while loading the switched capacitor array
* Set HSW to 01110111
* Set VSW to 11101110
* Set HSW and VSW to 00000000

It is possible to smooth the image further beyond a factor of 8. This may be performed by alternating between different 8-bit patterns for HSW or VSW. For example, smoothing horizontally over 16 pixels may be implemented by operating HSW as follows:

* Set HSW to 00000000 while loading the switched capacitor array
* Set HSW to 01111111
* Set HSW to 01110111
* Set HSW to 11110111
* Set HSW to 00000000.

Even further smoothing may be performed by repeating the sequence 01111111, 01110111, 11110111, 01110111 several more times. Very large quasi-smoothing kernels may be implemented in this manner.

Finally, it should be noted that entire rows or columns of pixels may be binned together by setting HSW or VSW to all ones.

Once the switched capacitor array has been operated to smooth the image, the capacitor values may be read out. It is only necessary to read out one value for each bin. For example, if binning down by a factor of 4 in each direction as shown above, every fourth row and column may be read out, which reduces the total number of pixels read out by a factor of 16. To read out the potential of row i and column j, set ROWSEL to i (if it is not already set to this value), set SELCAP to high (it may be left high while reading out the contents of the capacitor array), and set COLSEL to j.
8. Operation of individual components:
This section discusses the operation of individual components and modules of the chip. The instructions and commands are presented below in the general order of operation for a sample application, and with increased functionality towards the end of this section.

8.1 PIO12B:
Refer to Centeye documentation on the PIO12B interface for instructions on its use. The PIO12B interface is required for almost all communication with the chip. It should be noted that when WNR=0 and the chip is selected, the PIO12B will write the 8-bit ADC output to the IO0...IO7 lines and connect the buffered pixel signal to the ANALOG1 pad.

The notation "Send (XXX)" will be used to indicate that a 12-bit command XXX is sent to the vision chip, using the methods outlined in the PIO12B reference document. The value XXX may be an equation, for example "Send (ROWSEL+12)" indicates that command 0x800+12 (hex 0x800 plus decimal 12) is sent to the vision chip, which selects row 12.

The notation "XXX = Receive()" will be used to indicate an 8-bit value read from the vision chip, using the methods outlined in the PIO12B reference document.

The notation "Delay(XXX)" will be used to indicate that a delay of time XXX in microseconds should be performed.

8.2 Configuring the linear regulator:
Generally the first step performed is to configure the linear voltage regulator. A description of the voltage regulator's operation is beyond the scope of this document. Suffice it to say that to set up the linear voltage regulator to generate a desired VDDA, send the following four commands to the vision chip, in any order:

Send (LINREG1+0x24); // send first linear regulator command
Send (LINREG2+0x04); // send second linear regulator command
Send (LINREG3+X); // send third linear regulator command, where X is in the table below
    // the default for X is 0x06
Send (LINREG4+0x10); // send fourth linear regulator command, which enables the regulator

The value XXX may be chosen to select a specified output voltage using the following table. Note that in practice, voltages above 3V are preferred. Note that the voltage regulator is a low dropout linear regulator and cannot generate a higher voltage than VDD. For VDD=5V, we recommend setting the regulator to provide 4.46V using value 5.

(Table on following page.)

(Corrected from previous version)
<table>
<thead>
<tr>
<th>X</th>
<th>Resulting VDDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>1.96V</td>
</tr>
<tr>
<td>14</td>
<td>2.08V</td>
</tr>
<tr>
<td>13</td>
<td>2.23V</td>
</tr>
<tr>
<td>12</td>
<td>2.40V</td>
</tr>
<tr>
<td>11</td>
<td>2.60V</td>
</tr>
<tr>
<td>10</td>
<td>2.83V</td>
</tr>
<tr>
<td>9</td>
<td>3.12V</td>
</tr>
<tr>
<td>8</td>
<td>3.47V</td>
</tr>
<tr>
<td>7</td>
<td>3.75V</td>
</tr>
<tr>
<td>6</td>
<td>4.08V</td>
</tr>
<tr>
<td>5</td>
<td>4.46V</td>
</tr>
<tr>
<td>4</td>
<td>4.89V (not recommended)</td>
</tr>
<tr>
<td>3, 2, 1, and 0</td>
<td>Don't use</td>
</tr>
</tbody>
</table>
8.3 Bias generators and bias generator vault:

The Faraya series of vision chips include on-chip bias generators. These must be programmed and turned on before the analog portion of the chip functions e.g. before images can be acquired. **Note:** The two biases YUKNBIAS and NBIAS must be set to proper values or else the Faraya chip will draw large amounts of current!

Biases may be programmed using six-bit header and six-bit argument commands. Thus each bias may have a total of 64 different values. A resistor ladder is used to generate the biases. The actual bias voltages are generally a monotonic function of the setting value but are not linear. The PRSUPPLY bias can vary from about 40% to 100% of VDDA, with the value 0 corresponding to the lowest voltage and 63 corresponding to the highest voltage. The other biases can vary from about 0% to 50% of VDDA, with the value 0 corresponding to the highest voltage and 63 corresponding to the lowest voltage. **Note that for all biases other than PRSUPPLY, a higher setting corresponds to a lower voltage.**

To help prevent against accidental programming, some of the biases are placed behind a "vault". The vault must be "unlocked" before the biases can be set. Locking/unlocking the vault essentially controls whether the command signals are sent to the bias generators- when the vault is locked, the command signals do not make it through. The vault may be unlocked by setting the BIASSWITCH1, BIASSWITCH2, and BIASSWITCH3 registers to proper key values, and unlocked by setting any of these registers to a different value.

The chip output signal BIASSWITCH indicates the status of the vault, with a "1" being "unlocked" and a "0" being "locked". Monitoring the BIASSWITCH signal is a useful way to determine if the chip is receiving proper commands through the PIO12B interface. It is possible to override the BIASSWITCH signal, but in normal practice this should not be necessary.

Once the biases are set, the bias generators may be turned on by setting the CONNECTVDDA register with the appropriate key value. The CONNECTVDDA register is a vaulted register- the vault must be open to set this register. **Note:** Do not enable the CONNECTVDDA register until after the biases have been set. The default bias settings on power-up will draw a lot of current. The CONNECTVDDA signal is also output from the chip and may be monitored, with a "0" indicating "enabled" and a "1" indicating "disabled". Default is disabled.

The command sequence for 5V operation to open the vault, set the biases, turn on the bias generators, and close the vault are shown below, using recommended default values for the bias generators:

```
// Set the non-vaulted biases. These are needed for Faraya256plus and Faraya64plus only
Send (RESB+63); // Bottom of ADC range is set to lowest voltage
Send (REST+0);  // Top of ADC range is set to highest voltage
Send (YUKNBIAS+51); // This bias MUST be set before turning on bias generators
// Open the vault
Send (BIASSWITCH1+12); // 12 is first key
Send (BIASSWITCH2+21); // 21 is second key
Send (BIASSWITCH3+22); // 22 is third key
// Set the vaulted biases
Send (NBIAS1+56);
Send (NBIAS2+56);
Send (PRSUPPLY+63);
Send (VREF+63);
Send (ANALOGOUTBIAS+56);
```
// Turn on bias generators using CONNECTVDDA- The vault must be open to do this.
// DO NOT do this until AFTER you have set the biases, for this is what turn the bias
generators on and thus turns on the analog portion of the chip.
Send (CONNECTVDDA+2); // lowest three bits must be 010 to turn on biases
// Close the vault
Send (BIASSWITCH1+0); // Any value other than the key will close the vault
Send (BIASSWITCH2+0);
Send (BIASSWITCH3+0);

Notes:
1) The YUKNBIAS is the most critical bias and is not vaulted. This is an error on our behalf and will be
corrected in a future version of this chip. We apologize for this- you will just have to be careful.
2) The bias values should be set to their initial values before CONNECTVDDA is enabled.
3) It is not necessary to turn off the bias generators with CONNECTVDDA before adjusting bias voltages.
4) Again, note that PRSUPPLY is monotonic increasing with the bias setting, while all other biases are
monotonic decreasing.
5) The non-vaulted biases can be set whether the vault is open or closed.
8.4 Configuring and using the 8-bit flash ADC:

Registers:
RESB: Bias generator to generate lower reference voltage
REST: Bias generator to generate upper reference voltage
YUKNBIAS: Bias generator for Yukawa latch. **NOTE: this voltage cannot be too high a voltage / too low a value or else the chip will draw significant amounts of current!**

ADCOP: Configuration and operation of the ADC.
(Bit 0) YUKPHI: "0" to reset the ADC, "1" to initiate a conversion
(Bit 1) CONNECTADClN (legacy): Must be "0" for proper operation.
(Bit 2) SHORTRESB: When set to "1" this tends to short the RESB bias to GND. The effect is weak and thus this feature is best not used, e.g. just leave this value set to "0".

Basic mode: Operation via PIO12B

To operate the ADC via the PIO12B module, send the following commands. The variable pixel_value, which is an unsigned byte, will contain the digitized pixel value at the end of this sequence.

```c
// Note: RESB and REST commands need only be sent once
Send (RESB+63);  // This sets RESB to the lowest possible value. May be changed
Send (REST+0);   // This sets REST to the highest possible value. May be changed
// The remaining commands are sent once for each pixel conversion
Send (ADCOP+0);  // Reset ADC. Ideally do this before a new pixel value is selected
// <=< insert code to select a pixel by row and column here =>>
Send (ADCOP+1);  // Start ADC conversion
pixel_value = Receive(); // pixel_value will contain the 8-bit digitized pixel
```

Basic mode: Operation via the SHORTYUKPHI signal

Note: This feature does not work on the Faraya256plus vision chip. This is a silicon error on our behalf. Slightly faster operation may be obtained by using the SHORTYUKPHI signal, which is an input signal to the chip. Recall that this signal is internally pulled down. To use this mode, first the YUKPHI signal needs to be set high. Then the SHORTYUKPHI signal may be used to pull YUKPHI down. This may be performed as follows:

```c
// The following three commands need to be sent only once
Send (RESB+63);  // This sets RESB to the lowest possible value. May be changed
Send (REST+0);   // This sets REST to the highest possible value. May be changed
Send (ADCOP+1);  // Turn YUKPHI on. This only need to be performed once
// The following commands are sent once for each pixel conversion
SHORTYUKPHI = 1;  // Reset ADC. The processor must directly control this input signal
// <=< insert code to select a pixel by row and column here =>>
SHORTYUKPHI = 0;  // Start ADC conversion
pixel_value = Receive(); // pixel_value will contain the 8-bit digitized pixel
```
8.5 Reading out a single high resolution frame:

This section describes how to read out a single frame at high resolution, with no binning performed. The algorithm below assumes that no row amplification is used, and thus represents the simplest method. This algorithm will refer to code snippets already included above. Note that steps A through D need to be performed only once for setting up the chip (though there is no harm in sending them more than once), while step E is performed every frame.

// Step A: Set up biases (all chips)
// The first step is to set up the bias generators, as described above in Section 8.3
<Insert code here from Section 8.3>

// Step B: Configure the linear voltage regulator (Faraya256plus and Faraya64plus only)
// If we are using the linear voltage regulator to generate VDDA, then we initialize it here
// by setting the LINREG1 through LINREG4 registers, as described in Section 8.2
<Insert code here from Section 8.2>

// Step C: Configure ADC (Faraya256plus and Faraya64plus only)
// Since we have already set up RESB, REST, and YUKNBIAS, we only need to set
// up the ADCOP command register. For this example, we will be operating the ADC
// through the PIO12B interface, and not through the SHORTYUKPHI input signal
Send (ADCOP+0);       // Initialize ADC

// Step D: Setting up the focal plane (all chips)
// This next step sets up other various flags. Just follow them verbatim
Send (MISC+b'00111');  // SCSHORTCUT=0, VDDSWITCH=0, READPIX=1, LOADCAP=1, SELCAP=1
Send (MIDCONF+b'1100'); // SHORTCAP=1, SELUNITY=1, SELAMP=0
Send (MIDOP+b'000001'); // Turn off and disconnect amplifier
Send (VSW+0);            // No binning in vertical direction
Send (HSW+0);        // No binning in horizontal direction

// Step E: Read the image
// Now we simply loop through every row and column and digitize each pixel one at a time.
// The delay(x) function delays the program by x microseconds. These are suggested delays and
// may be tweaked for optimal performance. The x=ADC() command digitizes the analog signal
// at the ANALOG pin and stores the value in x.
For row = 0...63 do loop // for the Faraya256plus we loop through 0...255
    Send (ROWSEL+row); // Select row
    Delay(3);       // Delay- this delay value is suggested but not verified to be optimal
For column = 0...63 do loop // for Faraya256plus we loop through 0.255
    Send (ADCOP+0); // Reset ADC.
    Send (COLSEL+column); // Select column
    Delay(0.2);       // Delay- this value is suggested but not verified to be optimal
    // Below we digitize the pixel, using one of the methods discussed above
    // Option 1: (Faraya256plus and Faraya64plus) Use the on-chip ADC
    Send (ADCOP+1); // Start ADC conversion
    Delay(0.2);
    Set WNR = 0; // If it is not already 0
Set CS1 = 1; // enable chip output
pixel_value[row, column] = IO0...7 // Get digital pixel value from IO0...IO7 lines
Set CS1 = 0; // disable chip output
// Option 2: Digitize the analog signal generated by the ANALOG pin
Set WNR = 0; // if it is not already 0
Set CS1 = 1; // enable chip output
pixel_value[row, column] = ADC() // Digitize ANALOG signal and store
Set CS1 = 0; // disable chip output
end loop
end loop

Notes:
1) Note that it is not necessary to loop through every single pixel- one can skip rows and/or columns, and one can choose to read out just a block of pixels rather than the entire array.
2) The delay values above are suggested, but have not been verified to be optimal. In general a longer delay yields a better result, but slows down image acquisition.

8.6 Reading out a lower resolution image using binning
When generating a lower resolution image using binning, it is first necessary to load the capacitor array, then operate the VSW and HSW switches to perform binning, and then the pixel values may be read out and digitized. Steps A, B, and C are performed exactly the same as in the previous section. Steps D and E are modified as below, with step E split into three steps E1, E2, and E3. For this example, we are assuming that we are binning down the array by a factor of 8 in each direction, which turns a 64×64 array of pixels into an 8×8 array.

// Step D: Setting up the focal plane (all chips)
// This next step sets up other various flags. Just follow them verbatim
Send (MISC+b'00100'); // SCSHORTCUT=0, VDDSWITCH=0, READPIX=1, LOADCAP=0, SELCAP=0
Send (MIDCONF+b'1100'); // SHORTCAP=1, SELUNITY=1, SELAMP=0
Send (MIDOP+b'000001'); // Turn off and disconnect amplifier
Send (VSW+0); // No binning in vertical direction
Send (HSW+0); // No binning in horizontal direction

// Step E1: Load the capacitor array
For row = 0...63 do loop // for the Faraya256plus we loop through 0...255
    Send (ROWSEL+row); // Select row
    Delay(3); // Delay- this delay value is suggested but not verified to be optimal
    Send (MISC+b'00110'); // Set LOADCAP=1 to turn on "loadcap" for row.
        // This loads the row of capacitors with the current row value.
    Send (MISC+b'00100'); // Set LOADCAP=0 to turn off "loadcap" for row.
end loop
// Step E2: Binning
Send (VSW+b'11111110'); // Bin vertically
Send (HSW+b'01111111'); // Bin horizontally
Send (VSW+0); // Stop binning
Send (HSW+0);

// Step E3: Read the image
// This is similar to Step E above, with a minor change to the MISC register and
// the fact that we now only digitize every eight pixels
Send (MISC+b'00001'); // This allows "readcap" for each row to be turned on so
// that the row may be read out from the capacitor array
For row = 0,8,16,24,...,56 do loop // i.e. every eight row
    Send (ROWSEL+row); // Select row
    Delay(3); // Delay- this delay value is suggested but not verified to be optimal
    For column = 0,8,16,24,...,56 do loop // i.e. every eight column
        Send (ADCOP+0); // Reset ADC.
        Send (COLSEL+column); // Select column
        Delay(0.2); // Delay- this value is suggested but not verified to be optimal
        // Below we digitize the pixel, using one of the methods discussed above
        // Option 1: (Faraya256plus and Faraya64plus) Use the on-chip ADC
        Send (ADCOP+1); // Start ADC conversion
        Delay(0.2);
        Set WNR = 0; // If it is not already 0
        Set CS1 = 1; // enable chip output
        pixel_value[row,column] = IO0...7 // Get digital pixel value from IO0...IO7 lines
        Set CS1 = 0; // disable chip output
        // Option 2: Digitize the analog signal generated by the ANALOG pin
        Set WNR = 0; // If it is not already 0
        Set CS1 = 1; // enable chip output
        pixel_value[row,column] = ADC() // Digitize ANALOG signal and store
        Set CS1 = 0; // disable chip output
    end loop
end loop

Although the above algorithm has more steps than the simple algorithm, it will be much faster
since fewer pixels need to be digitized during the readout sequence.

Notes:
1) In the above example, one could read out for example rows 1,9,17 and so on rather than 0,8,16, and
   similarly for columns. This is fine as long as you consistently read out the same rows and columns so as
   to preserve the same fixed pattern noise, if using a fixed pattern noise compensation mask.
2) Not much delay is needed between steps for operating VSW and HSW. This happens very fast.
3) Generally if you want to bin an image twice in different ways, you need to reload the capacitor array
   with Step E1 first. However this can be skipped in certain cases, for example if you read out a raw
   resolution image, then binned down by a factor of two, and then binned down by a factor of 4.
4) Any smoothing performed with multiple switching steps is performed in Step E2.
8.7 Operation of row amplifiers:

[This part will be written in a later draft.]

8.8 Comments on Fixed Pattern Noise (FPN):

Fixed pattern noise (FPN) is inherent in all image sensor circuits. FPN refers to offsets between identically drawn pixel circuits that result from process variations. FPN manifests itself as a noise-like pattern that appears when an image sensor is provided with a uniform intensity. Ideally fixed pattern noise is measured once and then stored in memory. Then when a new frame is acquired, the stored FPN image is subtracted from the acquired image to produce a clean image.

Unfortunately, FPN is a function of pixel mode, amplification type, and binning methods. A different FPN mask will need to be acquired for each of these permutations.
9. Chip Command Codes:

Below is a list of all 12-bit commands, listed in increasing numerical order, and including unused command codes.

### Vaulted Biases

<table>
<thead>
<tr>
<th>Code</th>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>000 000 bbb bbb</td>
<td>NBIAS1</td>
<td>NBIAS1 vaulted bias (mid-type)</td>
</tr>
<tr>
<td>000 001 bbb bbb</td>
<td>VREF</td>
<td>VREF vaulted bias (mid-type)</td>
</tr>
<tr>
<td>000 010 bbb bbb</td>
<td>NBIAS2</td>
<td>NBIAS2 vaulted bias (mid-type)</td>
</tr>
<tr>
<td>000 011 bbb bbb</td>
<td>PRSUPPLY</td>
<td>PRSUPPLY vaulted bias (high-type)</td>
</tr>
<tr>
<td>000 100 bbb bbb</td>
<td>ANALOGOUTBIAS</td>
<td>ANALOGOUTBIAS vaulted bias (mid-type)</td>
</tr>
<tr>
<td>000 101 xxx xxx</td>
<td></td>
<td>(Not Used)</td>
</tr>
</tbody>
</table>

### Middle Amplifier Commands

<table>
<thead>
<tr>
<th>Code</th>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>000 110 xxc uax</td>
<td>MIDCONF</td>
<td>MIDCONF mid-amp configuration</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c: shortcap</td>
</tr>
<tr>
<td></td>
<td></td>
<td>u: selunity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>a: selamp</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NOTE bit 0 is unused</td>
</tr>
<tr>
<td>000 111 abc dpn</td>
<td>MIDOP</td>
<td>MIDOP mid-amp operation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>a: scref2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b: scin2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c: scref1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>d: scin1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>p: phi</td>
</tr>
<tr>
<td></td>
<td></td>
<td>n: nphi</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td>Enable Code</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>MISC</strong></td>
<td>Miscellaneous focal plane configuration</td>
<td>xxx010</td>
</tr>
<tr>
<td><strong>COLSEL</strong></td>
<td>Column select. Upper bits ignored</td>
<td>0x400</td>
</tr>
<tr>
<td><strong>ADCOP</strong></td>
<td>ADC operation (PLUS ONLY)</td>
<td>0x600</td>
</tr>
<tr>
<td><strong>RESB</strong></td>
<td>RESB non-vault bias (strong mid-type) (PLUS ONLY)</td>
<td>0x640</td>
</tr>
<tr>
<td><strong>REST</strong></td>
<td>REST non-vault bias (strong mid-type) (PLUS ONLY)</td>
<td>0x680</td>
</tr>
<tr>
<td><strong>YUKNBIAS</strong></td>
<td>YUKNBIAS non-vault bias (mid-type) (PLUS ONLY)</td>
<td>0x6C0</td>
</tr>
<tr>
<td><strong>CONNECTVDDA</strong></td>
<td>CONNECTVDDA (vaulted)</td>
<td>0x700</td>
</tr>
<tr>
<td><strong>BIASSWITCH1</strong></td>
<td>BIASSWITCH1</td>
<td>0x740</td>
</tr>
<tr>
<td><strong>BIASSWITCH2</strong></td>
<td>BIASSWITCH2</td>
<td>0x780</td>
</tr>
<tr>
<td><strong>BIASSWITCH3</strong></td>
<td>BIASSWITCH3</td>
<td>0x7C0</td>
</tr>
</tbody>
</table>
## Focal Plane Commands ROWSEL, HSW, and VSW

<table>
<thead>
<tr>
<th>Code</th>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>ROWSEL</td>
<td>ROWSEL Row select. Upper bits ignored.</td>
</tr>
<tr>
<td>101</td>
<td>HSW</td>
<td>HSW horizontal switches.</td>
</tr>
<tr>
<td>110</td>
<td>VSW</td>
<td>VSW vertical switches.</td>
</tr>
</tbody>
</table>

## Linear Regulator Commands (PLUS ONLY)

<table>
<thead>
<tr>
<th>Code</th>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>111 000</td>
<td>LINREG1</td>
<td>LINREG1 Linear Regulator configuration (PLUS ONLY)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>a: vrfb1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b: vrfb0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c: vrsb4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>d: vrsb3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>e: vrsb2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>f: vrsb1</td>
</tr>
<tr>
<td>111 001</td>
<td>LINREG2</td>
<td>LINREG2 Linear Regulator configuration (PLUS ONLY)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>a: vrsb8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b: vrsb7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c: vrsb6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>d: vrsb5</td>
</tr>
<tr>
<td>111 010</td>
<td>LINREG3</td>
<td>LINREG3 Linear Regulator configuration (PLUS ONLY)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>a: vrsw3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b: vrsw2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c: vrsw1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>d: vrsw0</td>
</tr>
<tr>
<td>111 011</td>
<td>LINREG4</td>
<td>LINREG4 Linear Regulator configuration (PLUS ONLY)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>a: vrdisc2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b: vrdisc1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c: vrsw18</td>
</tr>
<tr>
<td></td>
<td></td>
<td>d: vrsw6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>e: vrsw2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>f: vrsw1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>111 100</td>
<td>(Not Used)</td>
</tr>
<tr>
<td>111 101</td>
<td>(Not Used)</td>
</tr>
<tr>
<td>111 110</td>
<td>(Not Used)</td>
</tr>
<tr>
<td>111 111</td>
<td>(Not Used)</td>
</tr>
</tbody>
</table>
Erreta / Version Changes:

Version 1.0, February 15, 2011: Original Document

Disclaimer:

The information in this document is provided in connection with Centeye products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Centeye products. EXCEPT AS SET FORTH IN CENTEYE’S TERMS AND CONDITIONS OF SALE, CENTEYE ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL CENTEYE BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF CENTEYE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Centeye makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Centeye does not make any commitment to update the information contained herein. Centeye’s products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.