# Appendix 5.3.4
# Estimated star counts for autoguider

## Star Counts

The number of stars in an area of sky is given in Allen, Astrophysical Quantities table 117. The data below shows, (i) log(n) stars / square degree and (ii) number of stars / square arcsec for areas at a galactic latitude of 0 and 90 degrees.

| Mv | b=0 | | b=90 | | Mean | |
|---|---|---|---|---|---|---|
| | (i) | (ii) | (i) | (ii) | (i) | (ii) |
| 10 | 1.3 | 1.4E-06 | 0.6 | 2.7E-07 | 0.9 | 6.3E-07 |
| 11 | 1.7 | 4.1E-06 | 1.0 | 7.0E-07 | 1.3 | 1.7E-06 |
| 12 | 2.2 | 1.2E-05 | 1.3 | 1.6E-06 | 1.8 | 4.4E-06 |
| 13 | 2.6 | 3.1E-05 | 1.7 | 3.8E-06 | 2.2 | 1.1E-05 |
| 14 | 3.0 | 8.1E-05 | 2.0 | 7.9E-06 | 2.6 | 2.8E-05 |
| 15 | 3.4 | 2.0E-04 | 2.3 | 1.4E-05 | 2.9 | 6.7E-05 |
| 16 | 3.8 | 4.6E-04 | 2.5 | 2.7E-05 | 3.3 | 1.5E-04 |
| 17 | 4.1 | 1.0E-03 | 2.8 | 4.6E-05 | 3.6 | 3.4E-04 |

Table 1 Star counts at Galactic Latitudes
( i, log(n) stars / degree$^2$, ii, n stars / arcsec$^2$ )

## System

The autoguider detector will be a CCD mounted outside the dewar with a fixed pick-off mirror inside the dewar near the HAWAII detectors. A re-imaging lens mounted outside the dewar will produce an image on the autoguider CCD. A change of lens will allow the image resolution and field of view to be adjusted to match the telescope and observing conditions. The autoguider camera would also need a shutter.

### Bandwidth

Since the light to the autoguider does not pass through the science filters ( and cannot since it must be visible light ) any bandwidth is available for guiding. In practice it will be necessary to limit the bandwidth to reduce differential chromatic and refraction effects in the atmosphere. The sugested filter is from 600 nm to the CCD response cut-off at 1 µm.

### Focusing

The telescope will be focused onto the infrared detectors. The chromatic effects of the field corrector and atmosphere mean that the visible autoguider will be focused separately. If the relay lens is inside the dewar it may be technically simpler to move the CCD camera head to focus. Although the focus will probably not need changing after it is setup, this initial focus will be done while on the telescope and so must be driven remotely.

### Device

The visible CCD could be a Kodak CCD, with 1000 x 1500 x 9um pixels. With direct re-imaging optics this gives a field of view of 5.5 x 3.7 arcmin and a resolution of 0.2

arcsec/pixel. This matches well with the 18um pixels in the infrared camera. To obtain a larger field the reimaging lens could increase the image scale by a factor of 2 to match the seeing. The current design allows the whole CCD to fit in the unvignetted field with a 2:1 re-imaging.

The table below shows the number of stars expected in the field of view of the autoguider using the full CCD with a pixel scale of 0.2 arcsec/pixel.

| Mv | b=0 | | b=90 | | Mean | |
|---|---|---|---|---|---|---|
| | (i) | (ii) | (i) | (ii) | (i) | (ii) |
| 10 | 0.1 | 10 | 0.0 | 2 | 0.0 | 5 |
| 11 | 0.3 | 26 | 0.1 | 5 | 0.1 | 12 |
| 12 | 0.9 | 58 | 0.1 | 11 | 0.3 | 28 |
| 13 | 2.3 | 90 | 0.3 | 24 | 0.8 | 57 |
| 14 | 5.9 | 100 | 0.6 | 44 | 2.1 | 87 |
| 15 | 14.9 | 100 | 1.1 | 65 | 4.9 | 99 |
| 16 | 34.2 | 100 | 2.0 | 86 | 11.1 | 100 |
| 17 | 76.5 | 100 | 3.4 | 97 | 24.8 | 100 |

Table 2 Number of stars in A&G field
( i, Stars in field, ii, Percentage of fields with guide star available )

## Signal estimates

An estimate of how faint a star can be used for the autoguider.

Telescope = 2.5m ( INT )

Bandwidth = 0.3 µm

Exposure time = 0.5 s

Overall efficiency = 10%

The table below shows the detected photons/pixel/star with these parameters. The calculation assumes an image scale of 0.22 arcsec/pixel. In poor seeing the signal could be binned on-chip to reduce the effects of read noise with faint objects.

| Mv | Detected photons/pixel with seeing (") | | | |
|---|---|---|---|---|
| | 0.3 | 0.5 | 1.0 | 1.5 |
| 10 | 577186 | 144297 | 36074 | 16033 |
| 11 | 229782 | 57446 | 14361 | 6383 |
| 12 | 91478 | 22869 | 5717 | 2541 |
| 13 | 36418 | 9105 | 2276 | 1012 |
| 14 | 14498 | 3625 | 906 | 403 |
| 15 | 5772 | 1443 | 361 | 160 |
| 16 | 2298 | 574 | 144 | 64 |
| 17 | 915 | 229 | 57 | 25 |

Table 3 Detected photons / pixel with seeing

# Appendix 5.4.1
# Hawaii - AstroCam 4100 Multiplexor

## Aims:

Interface circuit board outside the dewar which connects the clock lines and power supplies from the CCD controller to the array and multiplexes the four chips / four quadrants of the camera onto the single input of the AstroCam 4100.

I. Multiplex any of 16 outputs onto the controller.

II. Keep full flexibility of operating any quadrants in each array.

III. Provide power supplies to HAWAII.

IV. Derive HAWAII address functions from CCD clock lines.

V. Simulate CCD reset for double correlated sampling

VI. Provide ESD protection to bias supplies where possible.

## HAWAII array

### Clocks

The HAWAII array requires four addressing clock lines FSYNC, LINE, LSYNC and PIXEL, and a RESET line which resets the currently selected line. There is also a READ enable line to each quadrant.

### Outputs

The four SOURCE outputs from each detector chip are connected directly to the multiplexor. These will be switched in turn to the signal processing chain.

### Bias

The HAWAII array requires 4 power supplies. Two are fixed +5v, analogue and digital power, and two are adjustable VRESET 0.5 - 1.0 V and BIASGATE 3.5-3.8 V.

The variable supplies are adjusted to change the well depth or working point of the output amplifier, they are only adjusted during array tests. On the telescope the supplies will not be adjusted and will be the same to all quadrants.

## Astrocam 4100 CCD Controller

### Clocks

The Astrocam 4100 CCD controller is designed to run 4 phase image transfer CCDs and so provides 12 clock lines ( 4 serial, 4 image, 4 store ) and a Reset line. The clock lines are transmitted as RS-422 levels.

### Bias

The 4100 system power supplies are generated on a board mounted near the dewar. These are produced from DACs driving amplifiers fed from +/- 5, 9, 15 and +25 volt supplies. The data for the DACS is multiplexed onto the same lines as the CCD clocks.

### Multiplexing

The camera has 16 separate outputs ( 4 chips * 4 quadrants ), the Astrocam 4100 is a single channel system. This system must allow one ( and only one ) output from the camera to be connected to the controller. The multiplex addressing scheme should be scaleable to allow operation with one chip or four ( or more ! ) and the possibility of using multiple controllers.

# Interface

## Addressing

The clock lines supplied from the controller are all equivalent in terms of slew rate and access speed.

The four HAWAII address functions FSYNC, PIXEL ,LSYNC ,LINE are supplied from the Image clocks, the TTL levels will be derived directly from the RS-422 receivers.

The CCD reset line will be used to control the double correlated sampling switch between the detector output and the dummy reference. This is closer to the CCD meaning of the reset line than using it to reset the pixel.

The read enable lines READ1-4 for each chip will be supplied from the CCD Store clocks. Four data bits are used to select four quadrants so that any pattern of quadrants within a chip can be selected, this allows the whole array to be reset in one pass. The chip select lines are produced form the same four clocks. The quadrant or chip address is latched by CHIP_SELECT and QUADRANT_SELECT lines driven from the ccd serial clocks 2 and 3.

The pixel RESET function is produced from CCD Serial clock 1. Only one line is needed since it is ANDed with the chip and quadrant select lines.

The READ and clock lines to each quadrant are ANDed with the chip address so that the commands reach only the quadrants in the intended chip. This also reduces clock line interference and power dissipation inside the dewar.

This arrangement allows any pattern of quadrants in any one chip to be reset, although only one chip can be reset at a time.

## Multiplexing

Only one of the 16 possible outputs of the camera will be connected to the controller at any time, this can be achieved with 4 switches or a 16:1 analogue multiplexor. The other outputs should be connected to ground, for both static safety and to reduce on-chip amplifier glow. This could be achieved using 8 relays. Four * 4PCO relays to select the chip, and four SPCO relays to select one of the four quadrants. Instead 16 SPCO relays are used to simplify the signal path, with one output being transmitted though only one relay. The unused outputs are all connected to ground. A logic gate disables the relay power if multiple quadrants or chips are selected since this implies a RESET operation.

## Supplies

The Astromed 3200 controller used for the NICMOS camera provided software programmable power supplies. In practice these were always used at the same setting and extra hardware was needed to ensure that they were never incorrectly set to the wrong value. The HAWAII system will not use programmable supplies but will generate levels needed directly from adjustable regulators ( LM317) fed from the fixed analogue supplies available at the head. All the supplies have smoothing capacitors. The two adjustable levels are produced from potential dividors driven from a regulated +5 supply. All the supplies to the chips are protected from out of range voltages with 5.6 V Zener diodes.

### Double Correlated Output

The CCD controller uses double correlated sampling system which measures the difference in signal between the CCD reset level and the charge being measured.

The HAWAII system will switch between the output source follower on the array and a reference level connected in parallel with the output transistor. They are both driven from the same supplies and so this will remove common mode noise. Since the system must switch between these levels in every read a high speed FET switch is needed. The reference level is produced by a potentiometer between AGND and the SOURCE bias. A single bit from the CCD reset line is used to control this switch.

### Electro-static protection

The pcb inside the dewar has static protection on the bias and power supplies, this is reinforced with more diode protection in the interface. The output sources are not protected inside the dewar since this may introduce noise or crosstalk, in the interface the output in use is biased the other outputs are grounded. The clock lines will be connected to CMOS multiplexor chips in the interface which provides some protection.

All lines which are not connected to the chip, such as those to detectors not being addressed must connect to ground. All lines to chips being accessed must be protected by diodes or large value resistors.

## Connections

### Astrocam4100

The Astrocam 4100 controller has a 60 pin IDC connector carrying the following functions.

| Pin | 4100 Camera | Function | Pin | 4100 Camera | Function |
|---|---|---|---|---|---|
| 1 | M_RST | CHIP ( REF\ ) | 31 | SYS_RESET | NC |
| 2 | M_RST\ | CHIP\ ( REF ) | 32 | SYS_RESET\ | NC |
| 3 | M_SER1 | RESET | 33 | PLSHUT | SHUTP |
| 4 | M_SER1\ | RESET\ | 34 | SHUTTER- | SHUT1 |
| 5 | M_SER2 | CHIP_SEL | 35 | PL25 | P25_1 |
| 6 | M_SER2\ | CHIP_SEL\ | 36 | SHUTTER-1 | SHUT2 |
| 7 | M_SER3 | QUAD_SEL | 37 | PL15 | P15_1 |
| 8 | M_SER3\ | QUAD_SEL\ | 38 | PL15 | P15_2 |
| 9 | M_IM1 | FSYNC | 39 | MI9 | M9_1 |
| 10 | M_IM1\ | FSYNC\ | 40 | MI9 | M9_2 |
| 11 | M_IM2 | LINE | 41 | PL9 | P9_1 |
| 12 | M_IM2\ | LINE\ | 42 | PL9 | P9_2 |
| 13 | M_IM3 | LSYNC | 43 | XPL5 | P5_1 |
| 14 | M_IM3\ | LSYNC\ | 44 | XPL5 | P5_2 |
| 15 | M_IM4 | PIXEL | 45 | MI15 | M15_1 |
| 16 | M_IM4\ | PIXEL\ | 46 | MI15 | M15_2 |
| 17 | M_ST1 | DATA1 | 47 | AGND | GND1 |
| 18 | M_ST1\ | DATA1\ | 48 | AGND | GND2 |
| 19 | M_ST2 | DATA2 | 49 | AGND | GND3 |
| 20 | M_ST2\ | DATA2\ | 50 | AGND | GND4 |
| 21 | M_ST3 | DATA3 | 51 | NC | NC |
| 22 | M_ST3\ | DATA3\ | 52 | NC | NC |

| | | | |
|---|---|---|---|
| 23 M_ST4 | DATA4 | 53 AGND | GND5 |
| 24 M_ST4\ | DATA4\ | 54 AGND | GND6 |
| 25 M_SPARE | NC | 55 AGND | GND7 |
| 26 M_SPARE\ | NC | 56 AGND | GND8 |
| 27 STROBE | NC | 57 PL15 | SENSP2 |
| 28 STROBE\ | NC | 58 SENS2 | SENS2 |
| 29 MODE | NC | 59 PL15 | SENSP1 |
| 30 MODE\ | NC | 60 SENS1 | SENS1 |

### Dewar plug

The connector to the dewar is a 25way D plug for each of the four chips.

| Pin | Function |
|---|---|
| 1 | FSYNC |
| 2 | LINE |
| 3 | LSYNC |
| 4 | PIXEL |
| 5 | AGND |
| 6 | APLUS |
| 7 | DGND |
| 8 | DPLUS |
| 9 | VRST |
| 10 | BIAS |
| 11 | NC |
| 12 | NC |
| 13 | NC |
| 14 | SOURCE1 |
| 15 | SOURCE2 |
| 16 | SOURCE3 |
| 17 | SOURCE4 |
| 18 | READ1 |
| 19 | READ2 |
| 20 | READ3 |
| 21 | READ4 |
| 22 | RESET |
| 23 | SAFETY GND |
| 24 | NC |
| 25 | NC |

## Changes

v1 Original form
v2 Changed order of clocks to ccd functions, doubt about spare/serial4
v3 Uses latch for chip/quadrant select, separate relay for each output.

# Appendix 5.5.1
# SAM code

Martin Beckett
Institute of Astronomy
-
U60
-
EPS448

$Id: hawaii.asm,v 1.3 1996/05/30 15:33:35 mgb Exp $

Astrocam 4100 SAM code for HAWAII chip
--------------------------------------
Version 1, Single quadrant, single chip

$Log: hawaii.asm,v $
;; Revision 1.3  1996/05/30  15:33:35  mgb
;; All bit patterns set
;;

Revision 1.2  1996/05/25 19:12:57  mgb
Empty defn for functions

Revision 1.1  1996/05/23 17:07:55  mgb
Initial revision

```
PART:
EPS448

% Define generic names for input pins %
INPUTS:
BIT0 @ 12
BIT1 @ 12
BIT2 @ 10
BIT3 @ 9
BIT4 @ 5
BIT5 @ 4
BIT6 @ 3
BIT7 @ 2

% Define alternative names for output pins %
OUTPUTS:
F0  @ 13        % RAMA0 / M_ST1(READ1)   %
F1  @ 14        % RAMA1 / M_ST2(READ2)   %
F2  @ 15        % RAMA2 / M_ST3(READ3)   %
F3  @ 16        % RAMA3 / M_ST4(READ4)   %
F4  @ 17        % RAMA4 / M_IM1(FSYNC)   %
```

```
F5  @ 18          % RAMA5 / M_IM2(LINE)       %
F6  @ 19          % HOLD  / M_IM3(LSYNC)      %
F7  @ 20          % RESI  / M_IM4(PIXEL)      %
F8  @ 22          % M_RES(CHIP) / OPEN1       %
F9  @ 23          % M_SER1(RESET) / OPEN2  %
F10 @ 24          % M_SER2(QUAD_SEL) / OPEN3       %
F11 @ 25          % M_SER3(CHIP_SEL) / OPEN4        %
F12 @ 26          % CONVERT / BLANK           %
F13 @ 27          % UPDOWN / FIELD   %
F14 @ 28          % MODE / LINE                    %
F15 @ 1           % latch select              %

% CONTROL (bit15 = 1) %
% RAMA0  variable address lsB            %
% RAMA1                         %
% RAMA2                         %
% RAMA3                         %
% RAMA4                         %
% RAMA5                   msB   %
% HOLD          hold DCS integrator H=freeze %
% RESI          reset integrator H=reset %
% CHIP          select real hawaii output (M_RES) %
% RESET reset HAWAII row, active low (M_SER1) %
% QUAD_SEL latches READx bits into head (M_SER2) %
% CHIP_SEL latches CHIPx bits into head (M_SER3) %
% CONVERT send ADC convert pulse, should be SER3 according to TechRef %
% UPDOWN  DCS integrator sign select H=up L=down %
% MODE    %

% CONTROL (bit15 = 0) %
% READ1 select quadrant/chip 1 (M_ST1) %
% READ2                    2 (M_ST2) %
% READ3                    3 (M_ST3) %
% READ4                    4 (M_ST4) %
% FSYNC vertical data bit (M_IM1) %
% LINE  vertical clock bit (M_IM2) %
% LSYNC horizontal data bit (M_IM3) %
% PIXEL horizontal clock bit (M_IM4) %
% OPEN1 not used %
% OPEN2 not used %
% OPEN3 not used %
% OPEN4 not used %
% BLANK pixel strobe to grabber board %
% FIELD field strobe to grabber board %
% LINE  line strobe to grabber board %




% General useful defn %
MACROS:
L = "0"           % Output line LOW %
```

```
H = "1"            % Output line HIGH %


%Addresses in the SAM %
ADDS_0  = "L L L L L L"        %Mode selection
                                0 - Initialize (A)
                                1 - RESET
                                2 - READ
                                3 - RESET-READ
                                4 - READ-RESET-READ
                                5 - QUAD Select
                                6 - CHIP Select
                                7 - Initialize (B)
                                %

ADDS_1  = "H L L L L L"   %Shutter open time - middle byte%
ADDS_2  = "L H L L L L"   %Shutter open time - lower byte%
ADDS_3  = "H H L L L L"   %Integrator integration time%
ADDS_4  = "L L H L L L"   %Rate parameter... (NOT USED)%
ADDS_5  = "H L H L L L"   %X offset of region (PXBEFORE_PAIRS)%
ADDS_6  = "L H H L L L"   %X size of region   (PXREAD_PAIRS)%
ADDS_7  = "H H H L L L"   %X after region     (PXAFTER_PARIS) %
ADDS_8  = "L L L H L L"   %Y offset of region (ROWBEFORE_PAIRS) %
ADDS_9  = "H L L H L L"   %Y size of region   (ROWREAD_PAIRS) %
ADDS_10 = "L H L H L L"   %Y after region     (ROWAFTER_PAIRS) %
ADDS_11 = "H H L H L L"   %Quadrant mask (was Vertical bin)%
ADDS_12 = "L L H H L L"   %Array mask (was Horizontal bin)%
ADDS_13 = "H L H H L L"   %NOT USED %
ADDS_14 = "L H H H L L"   %NOT USED %
ADDS_15 = "H H H H L L"   %NOT USED %
ADDS_16 = "L L L L H L"   %NOT USED %
ADDS_17 = "H L L L H L"   %Test routine pixel reads%
ADDS_18 = "L H L L H L"   %Shutter control %
ADDS_19 = "H H L L H L"   %X bin control (NOT USED) %
ADDS_20 = "L L H L H L"   %Shutter open time Top byte%
ADDS_21 = "H L H L H L"   %Shutter open delay
ADDS_22 = "L H H L H L"   %Shutter close delay
ADDS_23 = "H H H L H L"   %X size of quadrant (pairs) %
ADDS_24 = "L L L H H L"   %Y size of quadrant (pairs) %
ADDS_25 = "H L L H H L"   %Parallel pulse width (NOT USED)%
ADDS_26 = "L H L H H L"   %Parallel pulse delay (NOT USED)%
ADDS_27 = "H H L H H L"   %Reserved%
ADDS_28 = "L L H H H L"   %Reserved%
ADDS_29 = "H L H H H L"   %Reserved%
ADDS_30 = "L H H H H L"   %Reserved%
ADDS_31 = "H H H H H L"   %Reserved%


%States of the individual bits%
HOLD    = "L"   %Hold integrator%
_HOLD   = "H"   %Don't hold integrator%
RES_I   = "H"   %Reset integrator%
```

```
_RES_I  = "L"   %Don't reset integrator%
RES_T   = "H"   %Reset output transistor%
_RES_T  = "L"   %Don't reset output transistor%
CONVT   = "H"   %Do conversion%
_CONVT  = "L"   %No conversion%
UP      = "H"   %Integrate up%
DOWN    = "L"   %Integrate down%
WR_RM   = "H"   %Write to remote bus latches%
WR_CTRL = "L"   %Write to control bus latches%

% HAWAII renaming of some CCD functions %
CHIP    = "H"   %Select real output to mux (RES_T) %
REF     = "L"   %Select ref output = _CHIP  (_RES_T) %
CONTROL = "L"           %Latch to generally 4100 functions (WR_CTRL) %
CAMERA        = "H"   %Latch to camera head functions (WR_RM) %

NOSHUT  = "L L L L"  % All shutter bits off %


% For original data card, needed for PCI ? %
BLANK   = "H"
_BLANK  = "L"
FIELD   = "H"
_FIELD  = "L"
LINE    = "H"
_LINE   = "L"
NOSYNC  = "L L L" % Don't use sync lines%

% For programmable head %
PROG    = "L"   %Program DACs and latches%
CAMA    = "H"   %Send clocks through the remote head%

% HAWAII address clocks %
FSYNC   = "L" % Vertical data bit (active low) %
_FSYNC  = "H"
LINE    = "H"   % Vertical clock bit (double edge) %
_LINE  = "L"
LSYNC   = "L" % Horizontal data bit (active low) %
_LSYNC  = "H"
PIXEL  = "H"   % Horizontal clock bit (double edge) %
_PIXEL  = "L"
RESET = "L"    % Reset current row (active low) %
_RESET          = "H"


% Quadrant access %
READ1 = "H"    % Enable quadrant 1 %
_READ1          = "L"    % Disable quadrant 1 %
READ2 = "H"    % Enable quadrant 2 %
_READ2          = "L"    % Disable quadrant 2 %
READ3 = "H"    % Enable quadrant 3 %
_READ3          = "L"    % Disable quadrant 3 %
```

```
READ4= "H"    % Enable quadrant 4 %
_READ4         = "L"    % Disable quadrant 4 %

QUAD1          = "H L L L"      % Quadrant 1 only %
QUAD2          = "L H L L"      % Quadrant 2 only %
QUAD3          = "L L H L"      % Quadrant 3 only %
QUAD4          = "L L L H"      % Quadrant 4 only %
QUADALL = "H H H H"          % All quadrants active %
QUADNONE= "L L L L"     % No quadrants active %

CHIPALL = "H H H H"% All quadrants active %
CHIPNONE= "L L L L"     % No quadrants active %

DATAxxxx= "L L L L"     % Latch not active so don't care%
QUAD_SEL= "H"                 % Latch new quadrant mask %
CHIP_SEL= "H"                 % Latch new chip mask %
_SELECT = "L L"               % Don't change quad or chip select%

% Equation feature not used %
EQUATIONS:

% Programs, names based on RISC code %
PROGRAMS:

% ********************************************************************* %
% *                                                                   * %
% *  Astrocam based routines, initialisation                          *     %
% *                                                                   * %
% ********************************************************************* %


% ------------------------------------------------------------------ %
% Entry point at SAM reset                                               %
%         CCD and shutter settings removed                              %
% ------------------------------------------------------------------ %
0D:
  [ADDS_0 HOLD RES_I _RES_T _CONVT UP CAMA WR_RM] CONTINUE;
  [ BLANK FIELD LINE WR_CTRL ] JUMP MODE_SELECT;



% ------------------------------------------------------------------ %
% Test address 0 for mode                                                %
%                                                                        %
%   0 = 0 0 0 0  INIT_A                                                  %
%   1 = 1 0 0 0  RESET_ONLY                                              %
%   2 = 0 1 0 0  READ_ONLY                                               %
%   3 = 1 1 0 0  RESET_READ                                              %
%   4 = 0 0 1 0  READ_RESET_READ                                         %
%   5 = 1 0 1 0  not used                                                %
%   6 = 0 1 1 0  not used                                                %
%   7 = 1 1 1 0  INIT_B                                                  %
```

```
% -------------------------------------------------------------------- %
MODE_SELECT:
 IF /BIT0*/BIT1*/BIT2*/BIT3 THEN   %  0 0 0 0 %
         [ADDS_0 ] JUMP INIT_A;

 ELSEIF BIT0*/BIT1*/BIT2*/BIT3 THEN       %  1 0 0 0 %
         [ADDS_0 ] JUMP RESET_ONLY;

 ELSEIF /BIT0*BIT1*/BIT2*/BIT3 THEN       %  0 1 0 0 %
         [ADDS_0 ] JUMP READ_ONLY;

 ELSE
         [ADDS_0 ] JUMP OTHER_MODES;


OTHER_MODES:
 IF   BIT0*BIT1*/BIT2*/BIT3 THEN   %  1 1 0 0 %
         [ADDS_0 ] JUMP RESET_READ;

 ELSEIF  /BIT0*/BIT1*BIT2*/BIT3 THEN      %  0 0 1 0 %
         [ADDS_0 ] JUMP READ_RESET_READ;

 ELSEIF  BIT0*BIT1*BIT2*/BIT3 THEN        %  1 1 1 0 %
         [ADDS_0 ] JUMP INIT_B;

 ELSE
         [] JUMP END;  % PANIC %

% -------------------------------------------------------------------- %
% Read address and put value in counter %
% -------------------------------------------------------------------- %
READ_ADDRESS:
         [ ] CONTINUE;
         [ ] PUSHI;
         [ ] POPC;
         [ ] RETURN;


%------------------------------------------------------------------ %
% Initialise at power up A                                %
%------------------------------------------------------------------ %
INIT_A:
         %do a Reset_Read of a row to get black level %
         [ADDS_17 HOLD RES_I _RES_T _CONVT UP CAMA WR_RM] CALL
RESET_READ;



%------------------------------------------------------------------ %
% Initialise at power up B                                %
%------------------------------------------------------------------ %
INIT_B:
```

```
% *****************************************************************%
% *                                                               *%
% *   HAWAII setup routines                                     *%
% *                                                               *%
% *****************************************************************%

% ------------------------------------------------------------- %
% Select quadrant pattern                                       %
% ------------------------------------------------------------- %
QUADRANT:
        [ADDS_11 HOLD RES_I RES_T _RESET _SELECT UP CAMA WR_RM]
CONTINUE;
        [ADDS_11 HOLD RES_I RES_T _RESET _SELECT UP CAMA WR_RM]
PUSHI;
        [ADDS_11 HOLD RES_I RES_T _RESET _SELECT UP CAMA WR_RM] POPC;
        [ADDS_11 HOLD RES_I RES_T _RESET SELECT UP CAMA WR_RM]
RETURN;




%***************************************************************** %
% *                                                               * %
% *   HAWAII readout sub-routines                               * %
% *                                                                  * %
% ***************************************************************** %

% -------------------------------------------------------------- %
% Load row register for start of frame                          %
% -------------------------------------------------------------- %
FRAMESTART:
        [QUADxxxx _FSYNC _LINE _LSYNC _PIXEL NOSHUT NOSYNC]
CONTINUE;
        [QUADxxxx FSYNC  _LINE _LSYNC _PIXEL NOSHUT NOSYNC] CONTINUE;
        [QUADxxxx _FSYNC _LINE _LSYNC _PIXEL NOSHUT NOSYNC] RETURN;


% -------------------------------------------------------------- %
% Load registers for start of row                      %
%       need version for pos/neg rows                           %
% -------------------------------------------------------------- %
ROWSTART_ODD:

ROWSTART_EVEN:
```

```
%******************************************************************* %
% *                                                                * %
% *   HAWAII readout modes                                         *     %
% *                                                                * %
% ***************************************************************** %

% ---------------------------------------------------------------- %
% Reset Frame                                                      %
%         full frame only version                                  %
% ---------------------------------------------------------------- %
RESET_ONLY:
        [ ] RETURN;



% ---------------------------------------------------------------- %
% Read  Frame                                                      %
%         full frame only version                                  %
% --------------------------------------------------------------- %
READ_ONLY:
        [ ] RETURN;



% ---------------------------------------------------------------- %
% Reset_Read Frame                                                 %
%         full frame only version                                  %
% ---------------------------------------------------------------- %
RESET_READ:
        [ ] RETURN;



% ---------------------------------------------------------------- %
% Read_Reset_Read Frame                                                    %
%         full frame only version                                  %
% ---------------------------------------------------------------- %
READ_RESET_READ:
        [ ] RETURN;




PROGRAM:

%----------------------------
|                           |
| SAM RESET ON STARTUP POINT |
|                           |
 ----------------------------%
```

```
0D:
  [ADDS_0 HOLD  RES_I _RES_T H1  _CONVT UP   CAMA WR_RM] CONTINUE;
% Decide whether the hwinit routine or normal operation is required%
  [V0 CLOSE_SH  BLANK  FIELD  LINE WR_CTRL] JUMP MODE_SELECT;

%----------------------------
|                            |
| TEST ADDRESS ZERO FOR MODE |
|                            |
 ----------------------------%
MODE_SELECT:
 IF     /BIT0*/BIT1*/BIT2*/BIT3 THEN
  [ADDS_0   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] JUMP
INIT_DACS_A;
 ELSEIF  BIT0*/BIT1*/BIT2*/BIT3 THEN
  [ADDS_0   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] JUMP
FLUSH_ONLY;
 ELSEIF /BIT0* BIT1*/BIT2*/BIT3 THEN
  [ADDS_0   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] JUMP
EXPOSE_ONLY;
 ELSE
  [ADDS_0   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] JUMP
OTHER_MODES;

OTHER_MODES:
 IF     /BIT0*/BIT1* BIT2*/BIT3 THEN
  [ADDS_0   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] JUMP REGION;
 ELSEIF  BIT0*/BIT1* BIT2*/BIT3 THEN
  [ADDS_0   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] JUMP DUMMY;
 ELSEIF  BIT0* BIT1* BIT2*/BIT3 THEN
  [ADDS_0   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] JUMP
INIT_DACS_B;
 ELSE
  [ADDS_0   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] JUMP
FULL_FRAME;

%-------------------------------------
|                                     |
| INITIALISE THE DACS ON POWER UP (A) |
| (READS HALF PIXELS) (MODE 0)        |
|                                     |
 -------------------------------------%
INIT_DACS_A:
% Get CCD Xsize and Flush the serial register%
  [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL
SREG_FLUSH_F;
  [ADDS_17  HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CONTINUE;
  [ADDS_17  HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CONTINUE;
  [ADDS_17  HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] PUSHI;
  [ADDS_17  HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] POPC;
IDA_A:
```

```
 [ADDS_3   HOLD _RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL
HALF_PIXEL_READ;
 [ADDS_3   HOLD _RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] LOOPNZ
IDA_A;
IDA_END:
 [V0 CLOSE_SH   BLANK _FIELD  _LINE WR_CTRL] JUMP IDA_END;

%-------------------------------------
|                                    |
| INITIALISE THE DACS ON POWER UP (B) |
| (READS WHOLE PIXELS) (MODE 7)       |
|                                    |
 -------------------------------------%
INIT_DACS_B:
% Flush the WHOLE CCD%
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL
FLUSH_FRAME;
% Number of lines to read in setting black level%
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] LOADC
BLACK_V_READ;
IDB_A:
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL
VER_SHIFT_F;
% Number of pixels to read in each line%
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] PUSHLOADC
BLACK_H_READ;
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] JUMP
CHOOSE_RATE_7;
% Label for CHOOSE_RATE_7 to jump back to%
IDB_B:
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] POPC;
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] LOOPNZ
IDB_A;
IDB_END:
 [V0 CLOSE_SH   BLANK _FIELD  _LINE WR_CTRL] JUMP IDB_END;

%-------------------------------------
|                                    |
| TEST ADDRESS FOUR FOR RATE (MODE 7) |
|                                    |
 -------------------------------------%
CHOOSE_RATE_7:
 IF  BIT0*/BIT1*/BIT2*/BIT3 THEN
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL EPR1
RETURNTO IDB_B;
 ELSEIF /BIT0* BIT1*/BIT2*/BIT3 THEN
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL EPR2
RETURNTO IDB_B;
 ELSEIF  BIT0* BIT1*/BIT2*/BIT3 THEN
 [ADDS_19  HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL
EPR3_TEST RETURNTO IDB_B;
 ELSE
```

```
  [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL EPR0
RETURNTO IDB_B;

%----------------------
|                     |
| FLUSH FRAME (MODE 1) |
|                     |
 ----------------------%
FLUSH_ONLY:
  [V0 CLOSE_SH  BLANK  FIELD  LINE WR_CTRL] CALL FLUSH_FRAME;
EMPTY_FRAME_WAIT:
  [V0 CLOSE_SH  BLANK  FIELD _LINE WR_CTRL] JUMP EMPTY_FRAME_WAIT;

%-------------------------------------
|                                    |
| EMPTY CCD OF ALL ACCUMULATED CHARGE |
|                                    |
 -------------------------------------%
%This routine DOES NOT preserve the counter%
FLUSH_FRAME:
% Get the number of lines to flush (CCD Ysize)%
  [ADDS_24  HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL
READ_ADDRESS_F;
  [V0 CLOSE_SH  BLANK _FIELD  LINE WR_CTRL] CALL EIGHT_LINE_FLUSH_F;
  [V0 CLOSE_SH  BLANK _FIELD  LINE WR_CTRL] RETURN;

%----------------------------------------------
|                                             |
| FLUSH (N*8) LINES FROM THE PARALLEL REGISTER |
|                                             |
 ----------------------------------------------%
% Place N in the counter first
  This routine DOES NOT preserve the counter%
EIGHT_LINE_FLUSH_F:
  [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP  CAMA WR_RM] LOOPNZ
ELF_START
                                                          ONZERO ELF_EXIT;
ELF_START:
% Parallel bin EIGHT lines%
  [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP  CAMA WR_RM] PUSHLOADC
7D;
  [V0 CLOSE_SH  BLANK _FIELD  LINE WR_CTRL] CALL VER_SHIFT_F;
% Flush one horizontal line%
  [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP  CAMA WR_RM] CALL
SREG_FLUSH_F;
  [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP  CAMA WR_RM] POPC
                                                          GOTO EIGHT_LINE_FLUSH_F;
ELF_EXIT:
  [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP  CAMA WR_RM] RETURN;

%----------------------------------
|                                 |
```

```
| (N+1) VERTICAL SHIFTS OF CCD DATA |
| WITH FIELD ACTIVE (LOW)           |
|                                   |
 -----------------------------------%
% Place N in the counter first
  This routine DOES NOT preserve the counter%
VER_SHIFT_F:
% Save the counter (number of transfers) onto the stack %
  [ADDS_26  HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] PUSHLOADC
0D;
% Push the pulse widths and delays onto the stack%
  [V0 CLOSE_SH  BLANK _FIELD  LINE WR_CTRL] CONTINUE;
  [V0 CLOSE_SH  BLANK _FIELD  LINE WR_CTRL] PUSHI;
  [ADDS_25  HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CONTINUE;
  [V0 CLOSE_SH  BLANK _FIELD  LINE WR_CTRL] CONTINUE;
  [V0 CLOSE_SH  BLANK _FIELD  LINE WR_CTRL] PUSHI;
  [V0 CLOSE_SH  BLANK _FIELD  LINE WR_CTRL] PUSHI;
  [V0 CLOSE_SH  BLANK _FIELD  LINE WR_CTRL] PUSHI;
% Let's do it%
  [V1 CLOSE_SH  BLANK _FIELD  LINE WR_CTRL] POPC;
VSF_B:
  [V1 CLOSE_SH  BLANK _FIELD  LINE WR_CTRL] LOOPNZ VSF_B;
  [V2 CLOSE_SH  BLANK _FIELD  LINE WR_CTRL] POPC;
VSF_C:
  [V2 CLOSE_SH  BLANK _FIELD  LINE WR_CTRL] LOOPNZ VSF_C;
  [V1 CLOSE_SH  BLANK _FIELD  LINE WR_CTRL] POPC;
VSF_D:
  [V1 CLOSE_SH  BLANK _FIELD  LINE WR_CTRL] LOOPNZ VSF_D;
  [V0 CLOSE_SH  BLANK _FIELD  LINE WR_CTRL] POPC;
VSF_E:
  [V0 CLOSE_SH  BLANK _FIELD  LINE WR_CTRL] LOOPNZ VSF_E;
  [V0 CLOSE_SH  BLANK _FIELD  LINE WR_CTRL] POPC;
  [V0 CLOSE_SH  BLANK _FIELD  LINE WR_CTRL] LOOPNZ VER_SHIFT_F;
  [V0 CLOSE_SH  BLANK _FIELD  LINE WR_CTRL] RETURN;

%---------------------------------
|                                 |
| FLUSH THE WHOLE SERIAL REGISTER |
|                                 |
 ---------------------------------%
% This routine DOES NOT preserve the counter%
% FIELD is active %
SREG_FLUSH_F:
% Get the CCD Xsize%
  [ADDS_23  HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL
READ_ADDRESS_F;
% Flush the pixels%
  [V0 CLOSE_SH  BLANK _FIELD  LINE WR_CTRL] CALL EIGHT_PIXEL_FLUSH;
  [V0 CLOSE_SH  BLANK _FIELD  LINE WR_CTRL] RETURN;

%-------------------------------------------
|                                           |
```

```
| FLUSH (N*8) PIXELS FROM THE SERIAL REGISTER |
|                                             |
 --------------------------------------------%
% Place N in the counter first
  This routine DOES NOT preserve the counter%
EIGHT_PIXEL_FLUSH:
  [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP  CAMA WR_RM] LOOPNZ
EF_START
                                                          ONZERO EF_EXIT;
EF_START:
  [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP  CAMA WR_RM] PUSHLOADC
6D;
EF_A:
  [ADDS_4   HOLD  RES_I  RES_T H2 _CONVT UP  CAMA WR_RM] CONTINUE;
  [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP  CAMA WR_RM] LOOPNZ EF_A;
  [ADDS_4   HOLD  RES_I  RES_T H2 _CONVT UP  CAMA WR_RM] POPC
                                                          GOTO EIGHT_PIXEL_FLUSH;
EF_EXIT:
  [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP  CAMA WR_RM] RETURN;

%--------------------------
|                         |
| EXPOSE CCD ONLY (MODE 2) |
|                         |
 --------------------------%
EXPOSE_ONLY:
% Need to set ADDS_18 first %
  [ADDS_18 HOLD  RES_I _RES_T H1  _CONVT UP   CAMA WR_RM] CONTINUE;
EXPOSE_ONLY_WAIT:
  [V0 CLOSE_SH  BLANK  FIELD _LINE WR_CTRL] JUMP EXPOSE_ONLY_WAIT;

%-----------------------------
|                             |
| FULL FRAME READOUT (MODE 3) |
|                             |
 -----------------------------%
FULL_FRAME:
% Just go to the region routine rather than crashing %
  [ADDS_18  HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] JUMP REGION;


%-------------------------
|                         |
| REGION READOUT (MODE 4) |
|                         |
 -------------------------%
% This routine reads out a rectangular region of the CCD and flushes
  the rest of the area as quickly as possible
  Had to move the beginning and end of frame markers to the very beginning
  and very end of the frame because of the "take <n>" command%
REGION:
% Expose the CCD (check shutter type Uniblitz/Prontor) and flush output%
```

```
 [ADDS_18  HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL
EXPOSE;
% Clock VFG for start of frame%
 [V0 CLOSE_SH  BLANK _FIELD  LINE WR_CTRL] CALL CLOCK_VFG;
% Now starts the readout...
 Flush the first N*8 lines%
 [ADDS_8   HOLD  RES_I _RES_T H1  _CONVT UP   CAMA WR_RM] CALL
READ_ADDRESS_F;
 [V0 CLOSE_SH  BLANK _FIELD  LINE WR_CTRL] CALL EIGHT_LINE_FLUSH_F;
% Read N*8 line segments (FIELD IS ACTIVE)%
 [ADDS_9   HOLD  RES_I _RES_T H1  _CONVT UP   CAMA WR_RM] CALL
READ_ADDRESS_F;
 [V0 CLOSE_SH  BLANK _FIELD  LINE WR_CTRL] CALL
EIGHT_SEGMENT_READ;
% Flush the last N*8 lines%
 [ADDS_10  HOLD  RES_I _RES_T H1  _CONVT UP   CAMA WR_RM] CALL
READ_ADDRESS_F;
 [V0 CLOSE_SH  BLANK _FIELD  LINE WR_CTRL] CALL EIGHT_LINE_FLUSH_F;
% Clock VFG for end of frame%
 [V0 CLOSE_SH  BLANK  FIELD  LINE WR_CTRL] CALL CLOCK_VFG RETURNTO
REGION;

%----------------
|                |
| EXPOSE THE CCD |
|                |
 ----------------%
% Set ADDS_18 before calling this routine%
EXPOSE:
% One extra instruction to allow the address to settle properly%
 [ADDS_18  HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CONTINUE;
% Test whether the shutter action is actually required AND which type%
 IF /BIT0 * BIT1 * /BIT2 THEN
  % Open the Prontor shutter%
  [V0 OPEN_PR    BLANK  FIELD  LINE WR_CTRL] CONTINUE;
 ELSEIF /BIT0 * /BIT1 * BIT2 THEN
  % Open the Uniblitz shutter%
  [V0 OPEN_UN    BLANK  FIELD  LINE WR_CTRL] CONTINUE;
 ELSEIF /BIT0 * BIT1 * BIT2 THEN
  % Open another shutter%
  [V0 OPEN_LQ    BLANK  FIELD  LINE WR_CTRL] CONTINUE;
 ELSE
  % Just do the hold without open and close overheads%
  [V0 CLOSE_SH   BLANK  FIELD  LINE WR_CTRL] CALL
SHUTTER_OPEN_DELAY
                                            RETURNTO EXPOSE_EXIT;
% Get the shutter opening time%
 [ADDS_21  HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CONTINUE;
 [ADDS_21  HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CONTINUE;
 [ADDS_21  HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] PUSHI;
 [ADDS_21  HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] POPC;
% Wait around for the shutter to open%
```

```
EXPOSE_A:
  [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL
MS_DELAY;
  [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] LOOPNZ
EXPOSE_A;

% Check the type of shutter for hold pulse then wait%
  [ADDS_18  HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CONTINUE;
  [ADDS_18  HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CONTINUE;
  IF /BIT0 * BIT1 * /BIT2 THEN
   % Hold the Prontor shutter%
   [V0 HOLD_PR    BLANK  FIELD  LINE WR_CTRL] CALL
SHUTTER_OPEN_DELAY;
  ELSEIF /BIT0 * /BIT1 * BIT2 THEN
   % Hold the Uniblitz shutter%
   [V0 HOLD_UN    BLANK  FIELD  LINE WR_CTRL] CALL
SHUTTER_OPEN_DELAY;
  ELSE
   % Hold Liquid Crystal shutter%
   [V0 HOLD_LQ    BLANK  FIELD  LINE WR_CTRL] CALL
SHUTTER_OPEN_DELAY;

% Get the shutter closing time%
  [ADDS_22  HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CONTINUE;
  [ADDS_22  HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CONTINUE;
  [ADDS_22  HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] PUSHI;
  [ADDS_22  HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] POPC;
% Wait around for the shutter to close%
EXPOSE_D:
  [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL
MS_DELAY;
  [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] LOOPNZ
EXPOSE_D;
EXPOSE_EXIT:
% Flush the serial register%
  [V0 CLOSE_SH  BLANK  FIELD  LINE WR_CTRL] CALL SREG_FLUSH_F;
  [V0 CLOSE_SH  BLANK  FIELD  LINE WR_CTRL] RETURN;

%-----------------------------
|                            |
| DELAY WHILE SHUTTER IS OPEN |
|                            |
 -----------------------------%
% This routine DOES NOT preserve the counter%
SHUTTER_OPEN_DELAY:
% Delay for address to settle%
  [ADDS_20  HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CONTINUE;
  [ADDS_20  HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CONTINUE;
% Save highest byte on the stack%
  [ADDS_1   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] PUSHI;
  [ADDS_1   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CONTINUE;
% Save middle byte%
```

```
 [ADDS_2   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] PUSHI;
 [ADDS_2   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CONTINUE;
% Save bottom byte%
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] PUSHI;
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] POPC GOTO
SOD_C;
SOD_A:
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] PUSHLOADC
255D;
SOD_B:
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] PUSHLOADC
255D;
SOD_C:
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL
MS_DELAY;
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] LOOPNZ
SOD_C;
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] POPC;
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] LOOPNZ
SOD_B;
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] POPC;
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] LOOPNZ
SOD_A;
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] RETURN;

%--------------------------------------
|                              |
| SEND CLOCK PULSES TO HELP VFG CARD    |
| SO THAT IT CAN SORT ITSELF OUT.       |
|                              |
 --------------------------------------%
% This routine DOES preserve the counter%
CLOCK_VFG:
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] LOADC
VFG_PULSES;
CV_A:
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CONTINUE;
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CONTINUE;
 [ADDS_4   HOLD  RES_I _RES_T H1  CONVT UP   CAMA WR_RM] CONTINUE;
 [ADDS_4   HOLD  RES_I _RES_T H1  CONVT UP   CAMA WR_RM] LOOPNZ CV_A;
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] RETURN;

%--------------------------------------
|                              |
| READ OUT (N+1)*8 BINNED LINE SEGMENTS |
|                              |
 --------------------------------------%
% Place N in the counter first
 This routine DOES NOT preserve the counter
 Flushes unwanted pixels as fast as possible
 Uses parallel and serial binning%
EIGHT_SEGMENT_READ:
```

```
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] PUSHLOADC
7D;
ESR_A:
% Shift by the number of lines in ADDS_11%
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] PUSHLOADC
0D;
 [ADDS_11  HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL
READ_ADDRESS_F;
ESR_B:
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL
VER_SHIFT_F;
% Now starts the real readout...
 Dump pixels before active segment (6MHz)%
 [ADDS_5   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL
READ_ADDRESS_F;
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL
EIGHT_PIXEL_SLOW_FLUSH;
% Read out the pixels%
 [ADDS_6   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL
READ_ADDRESS_F;
% VFG  beginning of line (not the obvious place so that I can
 get the value on the inputs before LINE is asserted)%
 [V0 CLOSE_SH  BLANK _FIELD _LINE WR_CTRL] CONTINUE;
% Extra instruction to allow ADDS_4 to settle%
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CONTINUE;
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] JUMP
CHOOSE_RATE_4;
% This is a label for the CHOOSE_RATE_4 routine to jump back to%
ESR_C:
% Clock the VFG to indicate end of line%
 [V0 CLOSE_SH  BLANK _FIELD  _LINE WR_CTRL] CALL CLOCK_VFG;
 [V0 CLOSE_SH  BLANK _FIELD  LINE WR_CTRL] CALL CLOCK_VFG;
% Dump pixels after segment (6MHz)%
 [ADDS_7   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL
READ_ADDRESS_F;
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL
EIGHT_PIXEL_SLOW_FLUSH;
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] POPC;
% Go round again%
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] LOOPNZ
ESR_A;
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] POPC;
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] LOOPNZ
EIGHT_SEGMENT_READ;
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] RETURN;

%-------------------------------------------------
|                                                |
| SLOW FLUSH (N*8) PIXELS FROM THE SERIAL REGISTER |
|                                                |
 -------------------------------------------------%
% Place N in the counter first
```

```
  This routine DOES NOT preserve the counter%
EIGHT_PIXEL_SLOW_FLUSH:
  [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP  CAMA WR_RM] LOOPNZ
ESF_START
                                                              ONZERO ESF_EXIT;
ESF_START:
  [ADDS_4   HOLD  RES_I  RES_T H1 _CONVT UP  CAMA WR_RM] PUSHLOADC
6D;
ESF_A:
  [ADDS_4   HOLD  RES_I _RES_T H2 _CONVT UP  CAMA WR_RM] CONTINUE;
  [ADDS_4   HOLD  RES_I _RES_T H2 _CONVT UP  CAMA WR_RM] CONTINUE;
  [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP  CAMA WR_RM] CONTINUE;
  [ADDS_4   HOLD  RES_I  RES_T H1 _CONVT UP  CAMA WR_RM] LOOPNZ ESF_A;
  [ADDS_4   HOLD  RES_I _RES_T H2 _CONVT UP  CAMA WR_RM] CONTINUE;
  [ADDS_4   HOLD  RES_I _RES_T H2 _CONVT UP  CAMA WR_RM] POPC
                                                              GOTO
EIGHT_PIXEL_SLOW_FLUSH;
ESF_EXIT:
  [ADDS_4   HOLD _RES_I _RES_T H1 _CONVT UP  CAMA WR_RM] RETURN;

%------------------------------------
|                              |
| TEST ADDRESS FOUR FOR RATE (MODE 4) |
|                              |
 ------------------------------------%
CHOOSE_RATE_4:
 IF  BIT0*/BIT1*/BIT2*/BIT3 THEN
  [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL EPR1
RETURNTO ESR_C;
 ELSEIF /BIT0* BIT1*/BIT2*/BIT3 THEN
  [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL EPR2
RETURNTO ESR_C;
 ELSEIF  BIT0* BIT1*/BIT2*/BIT3 THEN
  [ADDS_19  HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL
EPR3_TEST RETURNTO ESR_C;
 ELSE
  [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL EPR0
RETURNTO ESR_C;

%--------------------
|                 |
| DUMMY MODE (MODE 5) |
|                 |
 --------------------%
% Could put a user routine here %
DUMMY:
  [V0 CLOSE_SH _BLANK  FIELD _LINE WR_CTRL] JUMP DUMMY;

%---------------------------------------------------------
|                                              |
| (N+1)*8 READS AT ONE CYCLE PER INTEGRATION (UNCORRELEATED) |
|                                              |
```

```
 ----------------------------------------------------------%
% Place N in the counter first
  This routine DOES NOT preserve the counter%
EPR0:
  [ADDS_4   HOLD _RES_I _RES_T H1 _CONVT DOWN CAMA WR_RM]
PUSHLOADC 6D
                                                            GOTO EPR0_B;
EPR0_A:
% Extra hold %
  [ADDS_4   HOLD _RES_I _RES_T H1 _CONVT DOWN CAMA WR_RM] CONTINUE;
  [ADDS_4   HOLD _RES_I  RES_T H2  CONVT DOWN CAMA WR_RM] PUSHLOADC
6D;
EPR0_B:
  [ADDS_4   HOLD  RES_I _RES_T H2 _CONVT DOWN CAMA WR_RM] CONTINUE;
  [ADDS_4  _HOLD _RES_I _RES_T H1 _CONVT DOWN CAMA WR_RM]
CONTINUE;
% Extra hold %
  [ADDS_4   HOLD _RES_I _RES_T H1 _CONVT DOWN CAMA WR_RM] CONTINUE;
  [ADDS_4   HOLD _RES_I  RES_T H2  CONVT DOWN CAMA WR_RM] LOOPNZ
EPR0_B;
  [ADDS_4   HOLD  RES_I _RES_T H2 _CONVT DOWN CAMA WR_RM] POPC;
  [ADDS_4  _HOLD _RES_I _RES_T H1 _CONVT DOWN CAMA WR_RM] LOOPNZ
EPR0_A;
  [ADDS_4   HOLD _RES_I  RES_T H1  CONVT DOWN CAMA WR_RM] RETURN;

%-------------------------------------------
|                                    |
| (N+1)*8 READS AT ONE CYCLE PER INTEGRATION |
|                                    |
 -------------------------------------------%
% Place N in the counter first
  This routine DOES NOT preserve the counter%
EPR1:
  [ADDS_4   HOLD _RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] PUSHLOADC
6D GOTO EPR1_B;
EPR1_A:
  [ADDS_4  _HOLD _RES_I _RES_T H1 _CONVT DOWN CAMA WR_RM]
PUSHLOADC 6D;
%Extra hold%
  [ADDS_4   HOLD _RES_I _RES_T H1 _CONVT DOWN CAMA WR_RM] CONTINUE;
  [ADDS_4   HOLD _RES_I  RES_T H2  CONVT DOWN CAMA WR_RM] CONTINUE;
EPR1_B:
  [ADDS_4   HOLD  RES_I _RES_T H2 _CONVT UP   CAMA WR_RM] CONTINUE;
  [ADDS_4  _HOLD _RES_I _RES_T H2 _CONVT UP   CAMA WR_RM] CONTINUE;
  [ADDS_4  _HOLD _RES_I _RES_T H1 _CONVT DOWN CAMA WR_RM]
CONTINUE;
%Extra hold%
  [ADDS_4   HOLD _RES_I _RES_T H1 _CONVT DOWN CAMA WR_RM] CONTINUE;
  [ADDS_4   HOLD _RES_I  RES_T H2  CONVT DOWN CAMA WR_RM] LOOPNZ
EPR1_B;
  [ADDS_4   HOLD  RES_I _RES_T H2 _CONVT UP   CAMA WR_RM] POPC;
```

```
  [ADDS_4  _HOLD _RES_I _RES_T H2 _CONVT UP   CAMA WR_RM] LOOPNZ
EPR1_A;
  [ADDS_4  _HOLD _RES_I _RES_T H1 _CONVT DOWN CAMA WR_RM]
CONTINUE;
%Extra hold%
  [ADDS_4   HOLD _RES_I _RES_T H1 _CONVT DOWN CAMA WR_RM] CONTINUE;
  [ADDS_4   HOLD _RES_I  RES_T H1  CONVT DOWN CAMA WR_RM] RETURN;

%-------------------------------------------
|                                  |
| (N+1)*8 READS AT TWO CYCLES PER INTEGRATION |
|                                  |
 -------------------------------------------%
% Place N in the counter first
  This routine DOES NOT preserve the counter%
EPR2:
  [ADDS_4   HOLD _RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] PUSHLOADC
6D
                                                    GOTO EPR2_B;
EPR2_A:
  [ADDS_4  _HOLD _RES_I _RES_T H1 _CONVT DOWN CAMA WR_RM]
PUSHLOADC 6D;
%Extra hold%
  [ADDS_4   HOLD _RES_I _RES_T H1 _CONVT DOWN CAMA WR_RM] CONTINUE;
  [ADDS_4   HOLD _RES_I  RES_T H2  CONVT DOWN CAMA WR_RM] CONTINUE;
EPR2_B:
  [ADDS_4   HOLD  RES_I _RES_T H2 _CONVT UP   CAMA WR_RM] CONTINUE;
  [ADDS_4  _HOLD _RES_I _RES_T H2 _CONVT UP   CAMA WR_RM] CONTINUE;
  [ADDS_4  _HOLD _RES_I _RES_T H2 _CONVT UP   CAMA WR_RM] CONTINUE;
  [ADDS_4  _HOLD _RES_I _RES_T H1 _CONVT DOWN CAMA WR_RM]
CONTINUE;
  [ADDS_4  _HOLD _RES_I _RES_T H1 _CONVT DOWN CAMA WR_RM]
CONTINUE;
%Extra hold%
  [ADDS_4   HOLD _RES_I _RES_T H1 _CONVT DOWN CAMA WR_RM] CONTINUE;
  [ADDS_4   HOLD _RES_I  RES_T H2  CONVT DOWN CAMA WR_RM] LOOPNZ
EPR2_B;
  [ADDS_4   HOLD  RES_I _RES_T H2 _CONVT UP   CAMA WR_RM] POPC;
  [ADDS_4  _HOLD _RES_I _RES_T H2 _CONVT UP   CAMA WR_RM] CONTINUE;
  [ADDS_4  _HOLD _RES_I _RES_T H2 _CONVT UP   CAMA WR_RM] CONTINUE;
  [ADDS_4  _HOLD _RES_I _RES_T H1 _CONVT DOWN CAMA WR_RM] LOOPNZ
EPR2_A;
  [ADDS_4  _HOLD _RES_I _RES_T H1 _CONVT DOWN CAMA WR_RM]
CONTINUE;
%Extra hold%
  [ADDS_4   HOLD _RES_I _RES_T H1 _CONVT DOWN CAMA WR_RM] CONTINUE;
  [ADDS_4   HOLD _RES_I  RES_T H1  CONVT DOWN CAMA WR_RM] RETURN;

%----------------------------------------------------
|                                            |
| (N+1)*8 READS AT THREE OR MORE CYCLES PER INTERGRATION |
| (NO BINNING)                                   |
```

```
|                                        |
 -------------------------------------------------------%
% Place N in the counter first
  This routine DOES NOT preserve the counter%
EPR3:
  [ADDS_3   HOLD _RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] PUSHLOADC
4D;
EPR3_A:
  [ADDS_3   HOLD _RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL
PIXEL_READ;
  [ADDS_3   HOLD _RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] LOOPNZ
EPR3_A;
  [ADDS_3   HOLD _RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL
PIXEL_READ;
  [ADDS_3   HOLD _RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] POPC;
  [ADDS_3   HOLD _RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL
PIXEL_READ;
  [ADDS_3   HOLD _RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] LOOPNZ
EPR3_B;
  [ADDS_3   HOLD _RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL
PIXEL_READ;
  [ADDS_3   HOLD _RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] RETURN;
EPR3_B:
  [ADDS_3   HOLD _RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL
PIXEL_READ
                                                           RETURNTO EPR3;

%-------------------------------------------------------
|                                        |
| (N+1)*8 READS AT THREE OR MORE CYCLES PER INTERGRATION |
| (BINNED BY TWO OR MORE)                          |
|                                        |
 -------------------------------------------------------%
% Place N in the counter first
  This routine DOES NOT preserve the counter%
EPR3_BINNED:
  [ADDS_3   HOLD _RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] PUSHLOADC
4D;
EPR3_BINNED_A:
  [ADDS_3   HOLD _RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL
BINNED_PIXEL_READ;
  [ADDS_3   HOLD _RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] LOOPNZ
EPR3_BINNED_A;
  [ADDS_3   HOLD _RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL
BINNED_PIXEL_READ;
  [ADDS_3   HOLD _RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] POPC;
  [ADDS_3   HOLD _RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL
BINNED_PIXEL_READ;
  [ADDS_3   HOLD _RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] LOOPNZ
EPR3_BINNED_B;
  [ADDS_3   HOLD _RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL
BINNED_PIXEL_READ;
```

```
  [ADDS_3   HOLD _RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] RETURN;
EPR3_BINNED_B:
  [ADDS_3   HOLD _RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL
BINNED_PIXEL_READ
                                                         RETURNTO EPR3_BINNED;

%-----------------------------------------
|                               |
| CHECK WHETHER SERIAL BINNING IS REQUIRED |
| (NOT USED IN FULL FRAME MODE)           |
|                               |
 -----------------------------------------%
% Test Xbin control address at the beginning of the line to see if
  binning is required.  Routine must be called with ADDR_19 set%
EPR3_TEST:
% Delay for address to settle%
  [ADDS_19   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CONTINUE;
  IF /BIT0 THEN
   [ADDS_3   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL EPR3;
  ELSE
   [ADDS_3   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL
EPR3_BINNED;
   [ADDS_3   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] RETURN;

%--------------------------------------------------------
|                                          |
| LOOP FOR READING ONE PIXEL OUT OF CCD USING THE INPUT    |
| LATCH INFORMATION FOR THE INTEGRATION TIME ON INTEGRATOR |
|                                          |
 --------------------------------------------------------%
% This routine DOES preserve the counter%
PIXEL_READ:
% First a dummy read to save the counter on the stack%
% Maybe could delete the first cycle%
  [ADDS_3   HOLD _RES_I  RES_T H2 _CONVT UP   CAMA WR_RM] PUSHLOADC
0D;
  [ADDS_3   HOLD  RES_I _RES_T H2 _CONVT UP   CAMA WR_RM] CONTINUE;
  [ADDS_3  _HOLD _RES_I _RES_T H2 _CONVT UP   CAMA WR_RM] PUSHI;
  [ADDS_3  _HOLD _RES_I _RES_T H2 _CONVT UP   CAMA WR_RM] POPC;
PR_A:
  [ADDS_3  _HOLD _RES_I _RES_T H2 _CONVT UP   CAMA WR_RM] LOOPNZ
PR_A;
  [ADDS_3  _HOLD _RES_I _RES_T H1 _CONVT DOWN CAMA WR_RM] PUSHI;
  [ADDS_3  _HOLD _RES_I _RES_T H1 _CONVT DOWN CAMA WR_RM] POPC;
PR_B:
  [ADDS_3  _HOLD _RES_I _RES_T H1 _CONVT DOWN CAMA WR_RM] LOOPNZ
PR_B;
  [ADDS_3   HOLD _RES_I _RES_T H1 _CONVT DOWN CAMA WR_RM] POPC;
  [ADDS_3   HOLD _RES_I _RES_T H1  CONVT DOWN CAMA WR_RM] RETURN;

%--------------------------------------------------------
|                                          |
```

```
| LOOP FOR READING ONE PIXEL OUT OF CCD USING THE INPUT    |
| LATCH INFORMATION FOR THE INTEGRATION TIME ON INTEGRATOR |
| (THIS ROUTINE INCLUDES BINNING)                          |
|                                                          |
 ----------------------------------------------------------%
% This routine DOES preserve the counter%
BINNED_PIXEL_READ:
% First a dummy read to save the counter on the stack%
% Maybe could delete this instruction%
  [ADDS_3   HOLD _RES_I  RES_T H2 _CONVT UP   CAMA WR_RM] PUSHLOADC
0D;
  [ADDS_3   HOLD  RES_I _RES_T H2 _CONVT UP   CAMA WR_RM] CONTINUE;
% Integrate on the reset%
  [ADDS_3  _HOLD _RES_I _RES_T H2 _CONVT UP   CAMA WR_RM] PUSHI;
  [ADDS_12 _HOLD _RES_I _RES_T H2 _CONVT UP   CAMA WR_RM] POPC;
BPR_A:
  [ADDS_12 _HOLD _RES_I _RES_T H2 _CONVT UP   CAMA WR_RM] LOOPNZ
BPR_A;
% Now do a loop for binning, always compresses by at least two%
  [ADDS_12  HOLD _RES_I _RES_T H2 _CONVT UP   CAMA WR_RM] PUSHI;
  [ADDS_3   HOLD _RES_I _RES_T H2 _CONVT DOWN CAMA WR_RM] POPC;
BPR_BIN:
  [ADDS_3   HOLD _RES_I _RES_T H1 _CONVT DOWN CAMA WR_RM] CONTINUE;
  [ADDS_3   HOLD _RES_I _RES_T H2 _CONVT DOWN CAMA WR_RM] LOOPNZ
BPR_BIN;
% Integrate on the accumulated charge%
  [ADDS_3  _HOLD _RES_I _RES_T H1 _CONVT DOWN CAMA WR_RM] PUSHI;
  [ADDS_3  _HOLD _RES_I _RES_T H1 _CONVT DOWN CAMA WR_RM] POPC;
BPR_B:
  [ADDS_3  _HOLD _RES_I _RES_T H1 _CONVT DOWN CAMA WR_RM] LOOPNZ
BPR_B;
  [ADDS_3   HOLD _RES_I _RES_T H1 _CONVT DOWN CAMA WR_RM] POPC;
  [ADDS_3   HOLD _RES_I _RES_T H1  CONVT DOWN CAMA WR_RM] RETURN;

%----------------------------------------------------------
|                                                          |
| LOOP FOR READING A HALF PIXEL OUT OF CCD USING THE INPUT |
| LATCH INFORMATION FOR THE INTEGRATION TIME ON INTEGRATOR |
|                                                          |
 ----------------------------------------------------------%
% This routine DOES preserve the counter%
HALF_PIXEL_READ:
% First a dummy read to save the counter onto the stack%
  [ADDS_3   HOLD _RES_I  RES_T H2 _CONVT UP   CAMA WR_RM] PUSHLOADC
0D;
  [ADDS_3   HOLD  RES_I _RES_T H2 _CONVT UP   CAMA WR_RM] CONTINUE;
  [ADDS_3  _HOLD _RES_I _RES_T H2 _CONVT UP   CAMA WR_RM] PUSHI;
  [ADDS_3  _HOLD _RES_I _RES_T H2 _CONVT UP   CAMA WR_RM] POPC;
HPR_A:
  [ADDS_3  _HOLD _RES_I _RES_T H2 _CONVT UP   CAMA WR_RM] LOOPNZ
HPR_A;
% Read convertor in the middle of this pixel%
```

```
 [ADDS_3   HOLD _RES_I _RES_T H2 _CONVT UP   CAMA WR_RM] POPC;
 [ADDS_3   HOLD _RES_I _RES_T H2  CONVT UP   CAMA WR_RM] RETURN;

%----------------------------
|                     |
| DELAY FOR ONE MILLISECOND |
|                     |
 ---------------------------%
% This routine DOES preserve the counter%
MS_DELAY:
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] PUSHLOADC
99D;
MS_A:
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] CALL
US10_DELAY;
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] LOOPNZ MS_A;
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] POPC;
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] RETURN;

%---------------------------------------------------------
|                                          |
| DELAY FOR TEN MICROSECONDs (-2 clocks for calling loop) |
|                                          |
 --------------------------------------------------------%
% This routine DOES preserve the counter%
% Waggles the serial clocks and reset at 200kHz%
US10_DELAY:
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] PUSHLOADC
116D;
US10_A:
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] LOOPNZ
US10_A;
 [ADDS_4   HOLD  RES_I  RES_T H2 _CONVT UP   CAMA WR_RM] LOADC 116D;
US10_B:
 [ADDS_4   HOLD  RES_I  RES_T H2 _CONVT UP   CAMA WR_RM] LOOPNZ
US10_B;
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] POPC;
 [ADDS_4   HOLD  RES_I _RES_T H1 _CONVT UP   CAMA WR_RM] RETURN;

%-----------------------------------------------------------------
|                                                |
| READ ADDRESS INFORMATION AND RETURN IN THE COUNTER            |
| (FIELD ACTIVE [LOW], LINE INACTIVE [HIGH] AND SHUTTER CLOSED)   |
|                                                |
 ----------------------------------------------------------------%
READ_ADDRESS_F:
 [V0 CLOSE_SH   BLANK _FIELD  LINE WR_CTRL] CONTINUE;
 [V0 CLOSE_SH   BLANK _FIELD  LINE WR_CTRL] PUSHI;
 [V0 CLOSE_SH   BLANK _FIELD  LINE WR_CTRL] POPC;
 [V0 CLOSE_SH   BLANK _FIELD  LINE WR_CTRL] RETURN;
```

# Appendix 5.2.2
# Filter specifications

## Filters

The CIRSI camera uses separate filters in front of each of the four detectors. This allows the use of small 32 mm square filters rather than a large expensive single element. The same filters are used in the test dewars and the COHSI spectrograph.

Standard J and H filters have been ordered from BARR associates and delivery is expected now.

| Filter | Central Wavelength | Bandwidth (50%) | Peak Transmission |
|---|---|---|---|
| J | 1.25 | 1.12 - 1.40 | 80 % |
| H | 1.65 | 1.48 - 1.82 | 90 % |