ASX340AT Developer Guide

1/4−Inch VGA System−on−a−Chip (SOC) CMOS Digital Image Sensor

AND9270/D

About this Document

This developer guide is a reference for hardware and software engineers developing camera systems using the **onsemi**'s ASX340AT. The ASX340AT is a complete system−on−a−chip (SOC) image sensor that integrates seamlessly in today's automotive applications. It incorporates sophisticated on−chip camera functions and is programmable through a serial interface. This document provides information on hardware interfaces, camera control, and register programming recommendations to optimize image quality. The starting point for all tuning is using **onsemi** recommended settings, which are loaded when the Demo Initialization preset is run within DevWare.

The ASX340AT data sheet, register reference, and Host Command Interface documents should be used along with this guide as a reference for specific register and programming information.

Conventions and Notations

This developer guide follows the conventions and notations described below.

- Hexadecimal numbers have 0x prefix
- Binary numbers have 0b prefix
- Example: 0b1010 = 0xA
- Fixed point notation
- 0.8 (0.0 through 254/255)
- 1.7 (0.0 through 1 + 127/128)
- Signed fixed point notation –8.0 (0 through 0xF800)
- I/O signals can be LOW (0 or D_{GND}), HIGH (1 or V_{DD}_IO), or floating (high impedance or High−Z)
- Timing diagrams are not drawn to scale and do not illustrate the actual number of clocks necessary.

TABLE OF CONTENTS

GENERAL DESCRIPTION

The **onsemi** ASX340AT is a VGA−format, single−chip CMOS active−pixel digital image sensor for automotive applications. It captures high−quality color images at VGA resolution and outputs NTSC or PAL interlaced composite video.

The VGA CMOS image sensor features **onsemi**'s breakthrough low−noise imaging technology that achieves superior image quality (based on signal−to−noise ratio and low−light sensitivity) while maintaining the inherent size, cost, low power, and integration advantages of **onsemi**'s advanced active pixel CMOS process technology.

The ASX340AT is a complete camera−on−a−chip. It incorporates sophisticated camera functions on−chip and is programmable through a simple two−wire serial interface or by an attached SPI EEPROM or Flash memory that contains setup information that may be loaded automatically at startup.

The ASX340AT performs sophisticated processing functions including color recovery, color correction, sharpening, programmable gamma correction, auto black reference clamping, auto exposure, 50 Hz/60 Hz flicker detection and avoidance, lens shading correction, auto white balance (AWB), and on−the−fly defect identification and correction.

The ASX340AT outputs interlaced−scan images at 60 or 50 Fields per Second, supporting both NTSC and PAL video formats. It supports progressive output as well. The image data can be output on one of two output ports:

- Composite analog video (single−ended and differential output support)
- Parallel 8−, 10−bit digital

THERMAL RESISTANCE OF IBGA/ CSP PACKAGE

The thermal resistance values here are provided for reference only.

Table 1. THERMAL RESISTANCE OF IBGA/ CSP PACKAGE (°C/W)

Test Board	Parameters	IBGA	CSP
1SOP Test Board	θ_{JA}	64.72	61.27
	θ_{JB}	19.45	11.35
2S2P Test Board	θ_{JA}	36.57	31.51
	θ_{JB}	18.09	10.57
Theta JC	θ_{JC} Bottom	10.85	4.95
	θ_{JC} Top Glass	50.33	27.51
	θ_{JC} Top Glass and Encap	25.56	N/A (Note 7)

NOTE: Device will meet the specifications after thermal equilibrium has been established when mounted in a test socket or printed circuit board with maintained transverse airflow greater than 500 lfpm.

1. Chip Power: 236 mW.
2. Ambient temperature: 25°C.
3. Junction temperature: the center temperature of silicon die.
4. 1S0P PCB: JEDEC standard JESD51−9, metal trace on top of PCB only, with metal finish thickness of 70 μm.
5. 2S2P PCB: JEDEC standard JESD51−9, besides the metal trace on top, two copper planes (with 35 μm thickness) are embedded in the PCB.
6. Theta JC: heat sink with 25°C is mounted on bottom of the solder balls or top of the encapsulant or/and glass (excluding optical area).
7. There is no encapsulation in the CSP package.

SYSTEM POWER CONFIGURATION

For low−noise operation, the ASX340AT requires separate power supplies for analog and digital. Incoming digital and analog ground conductors need to be separated in the imager module. Both power supply rails should be decoupled from ground using capacitors as close as possible to the die. The use of inductance filters is not recommended on the power supplies or output signals.

Table 2. ASX340AT POWER PAD INFORMATION

Pin Number	Voltage Name	Voltage	Description
A5, A7, D8, E7, G1, G3	V_{DD}	1.8 V	Digital core logic. Can be connected to the output of the transistor of the off−chip bypass transistor or an external 1.8 V
B2, B8, C8, E3, E8, G8, H8	V_{DD}_IO	2.8 V	Digital Input/Output
H5	V_{DD}_DAC	2.8 V	Video DAC
A8	V_{DD}_PLL	2.8 V	PLL
B4, H6	V_{AA}	2.8 V	Analog circuitry
H7	V_{AA}_PIX	2.8 V	Analog pixel array. Must be at the same voltage potential as V_{AA}

ARCHITECTURE OVERVIEW

Internal Block Diagram

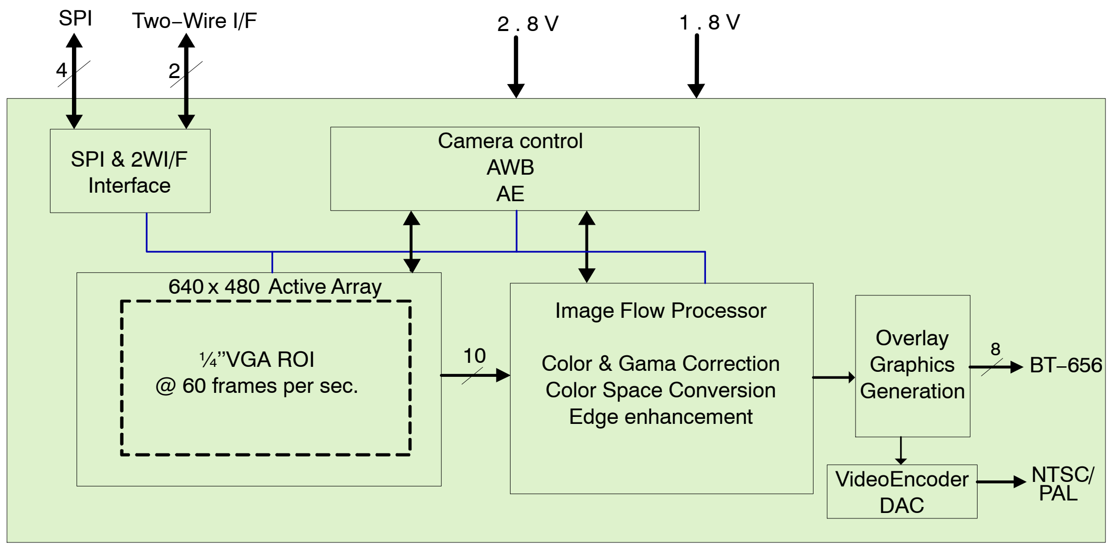


Figure 1. Internal Block Diagram

System Block Diagram

The system block diagram will depend on the application. The system block diagram in Figure 2 shows all components; optional peripheral components are highlighted. Control information will be received by a micro−controller through the automotive bus to communicate with the ASX340AT through its two−wire serial bus. Optional components will vary by application.

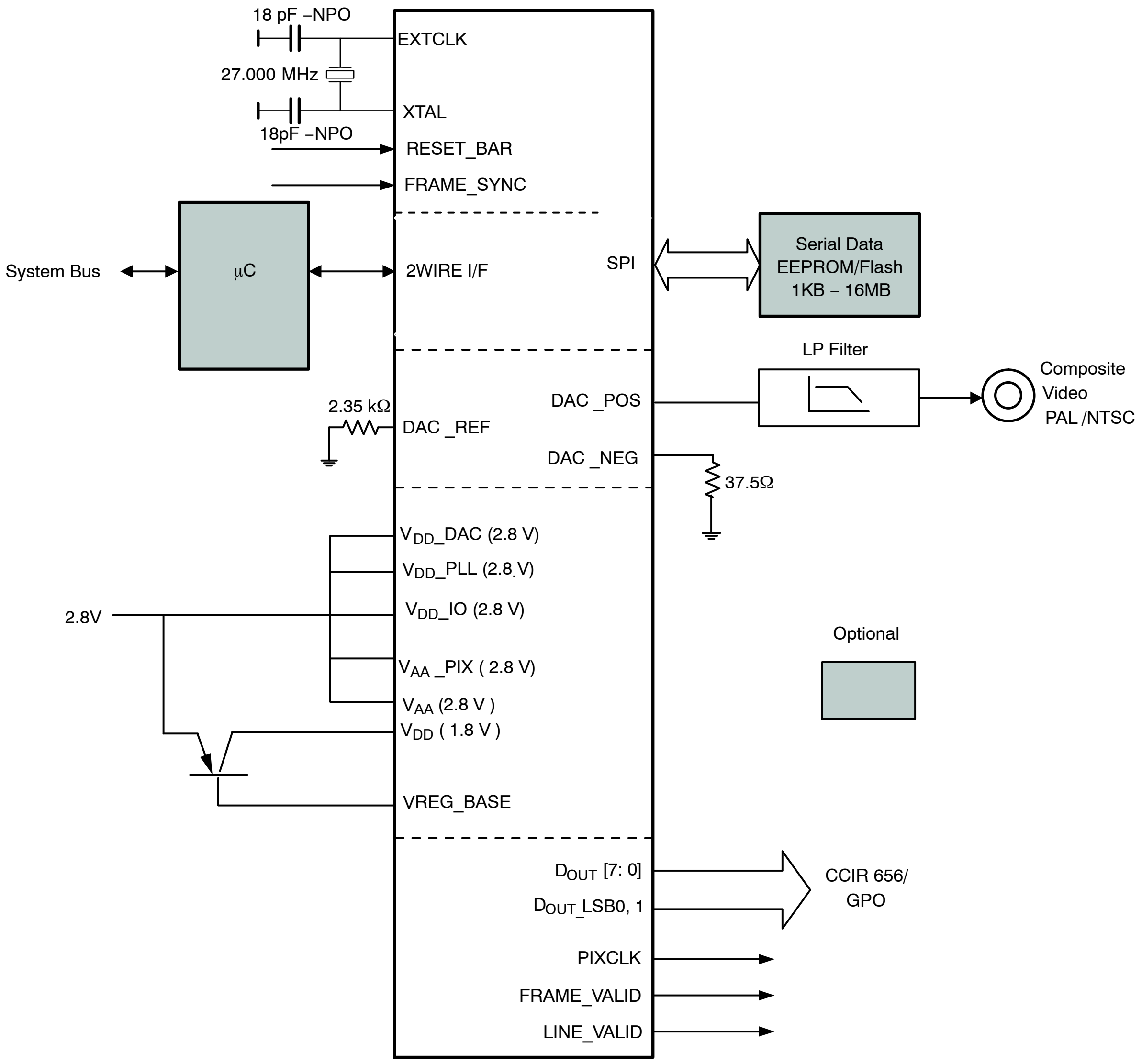


Figure 2. System Block Diagram

Crystal Usage

As an alternative to using an external oscillator, a fundamental 27 MHz crystal may be connected between EXTCLK and XTAL. Two small loading capacitors of 10–22 pF of NPO dielectric should be added as shown in Figure 3.

onsemi does not recommend using the crystal option for applications above 85°C. A crystal oscillator with temperature compensation is recommended.

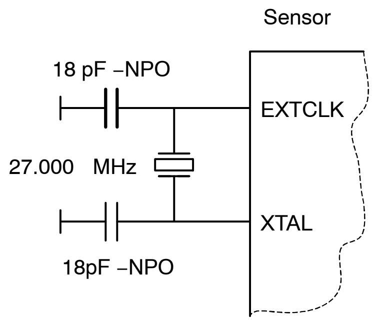


Figure 3. Using a Crystal Instead of an External Oscillator

1. Value of load capacitor is crystal dependent. Crystal with small load capacitor is recommended.

SCHEMATIC DESIGN GUIDELINES

This section is intended to provide guidelines on circuit design for ASX340AT sensors. This includes how to connect each pin, when used or not used, as well as V_{DD} internal regulator usage, and other design requirements.

Pin Assignments and Descriptions

Table 3 summarizes the pin assignments of all the critical interfacing pins. Pin 1 is not populated with a ball. That allows the device to be identified by an additional marking.

Table 3. PIN ASSIGNMENTS

	1	2	3	4	5	6	7	8
A		EXTCLK	GPIO13	DAC_REF	VDD	DOUT0	VDD	VDD_PLL
B	XTAL	VDD_IO	DOUT_LSB1	VAA	GND	DOUT2	DOUT1	VDD_IO
C	GPIO12	DOUT_LSB0	TCK	TRST_N	GND	DOUT4	DOUT3	VDD_IO
D	GND	RESET_BAR	TDI	SPI_SCLK	GND	DOUT6	DOUT5	VDD
E	FRAME_SYNC	SADDR	VDD_IO	SPI_SDI	AGND	PIXCLK	VDD	VDD_IO
F	SCLK	SDATA	TMS	AGND	DAC_POS	ATEST2	FRAME_VALID	DOUT7
G	VDD	TDO	VDD	VREG_BASE	DAC_NEG	ATEST1	LINE_VALID	VDD_IO
H	GND	SPI_CS_N	SPI_SDO	VPP	VDD_DAC	VAA	VAA_PIX	VDD_IO

A detailed description of each pin is provided in Table 4. The Description column provides critical instructions on what kind of signals should be connected to each pin. For critical information like whether a pin can be left unconnected/floating or has to be tied to the ground during normal operation and/or not used, it is specified also in the Description column. Please also refer to the Note column in Table 8 on pin connection requirements.

Table 4. PIN DESCRIPTION

Pin Number	Pin Name	Type	Description
Clock and Reset			
A2	EXTCLK	Input	Master input clock (27 MHz by default): This can either be a square−wave generated from an oscillator (in which case the XTAL input must be left unconnected) or connected directly to a crystal.
B1	XTAL	Output	If EXTCLK is connected to one pin of a crystal, this signal is connected to the other pin; otherwise this signal must be left unconnected.
D2	RESET_BAR	Input	Asynchronous active−low reset: When asserted, the device will return all interfaces to their reset state. When released, the device will initiate the boot sequence. This signal has an internal pull−up resistor.
E1	FRAME_SYNC	Input	This input can be used to set the output timing of the ASX340AT to a fixed point in the frame. The input buffer associated with this input is permanently enabled. This signal must be connected to GND if not used.
Register Interface			
F1	SCLK	Input	These two signals implement the serial communications protocol for access to the internal registers and variables.
F2	SDATA	Input/Output	
E2	SADDR	Input	This signal controls the device ID that will respond to serial communication commands. This signal must be permanently tied to VDD_IO or GND to determine which device ID is selected as described below. Two−wire serial interface device ID selection: 0: 0x90 1: 0xBA
SPI Interface			
D4	SPI_SCLK	Output	Clock output for interfacing to an external SPI memory such as Flash/EEPROM. Tristate when RESET_BAR is asserted.
E4	SPI_SDI	Input/Output	Data in from SPI device. This signal has an internal pull−up resistor.
H3	SPI_SDO	Output	Data out to SPI device. Tristate when RESET_BAR is asserted.
H2	SPI_CS_N	Output	Chip selects to SPI device. Tristate when RESET_BAR is asserted.
(Parallel) Pixel Data Output			
F7	FRAME_VALID	Input/Output	Pixel data from the ASX340AT can be routed out on this interface and processed externally. To save power, these signals are driven to a constant logic level unless the parallel pixel data output or alternate (GPIO) function is enabled for these pins. For more information see Table 9. This interface is disabled by default. The slew rate of these outputs is programmable. These signals can also be used as general purpose input/outputs.
G7	LINE_VALID	Input/Output	
E6	PIXCLK	Output	
F8, D6, D7, C6, C7, B6, B7, A6	DOUT[7:0]	Output	
B3	DOUT_LSB1	Input/Output	When the sensor core is running in bypass mode, it will generate 10 bits of output data per pixel. These two pins make the two LSB of pixel data available externally. Leave DOUT_LSB1 and DOUT_LSB0 unconnected if not used. To save power, these signals are driven to a constant logic level unless the sensor core is running in bypass mode or the alternate function is enabled for these pins. For more information see Table 16, “GPIO Bit Descriptions,”. The slew rate of these outputs is programmable.
C2	DOUT_LSB0	Input/Output	

Table 4. PIN DESCRIPTION (continued)

Pin Number	Pin Name	Type	Description
Composite Video Output			
F5	DAC_POS	Output	Positive video DAC output in differential mode. Video DAC output in single−ended mode. This interface is enabled by default using NTSC/PAL signaling. For applications where composite video output is not required, the video DAC can be placed in a power−down state under software control.
G5	DAC_NEG	Output	Negative video DAC output in differential mode.
A4	DAC_REF	Output	External reference resistor for the video DAC.
Manufacturing Test Interface			
D3	TDI	Input	JTAG Test pin (Reserved for Test Mode)
G2	TDO	Output	JTAG Test pin (Reserved for Test Mode)
F3	TMS	Input	JTAG Test pin (Reserved for Test Mode)
C3	TCK	Input	JTAG Test pin (Reserved for Test Mode)
C4	TRST_N	Input	Connect to GND.
G6	ATEST1	Input	Analog test input. Connect to GND in normal operation.
F6	ATEST2	Input	Analog test input. Connect to GND in normal operation.
GPIO			
C1	GPIO12	Input/Output	Dedicated general−purpose input/output pin.
A3	GPIO13	Input/Output	Dedicated general−purpose input/output pin.
Power			
G4	VREG_BASE	Supply	Voltage regulator control. Leave floating if not used.
A5, A7, D8, E7, G1, G3	VDD	Supply	Supply for VDD core: 1.8 V nominal. Can be connected to the output of the transistor of the off−chip bypass transistor or an external 1.8 V power supply.
B2, B8, C8, E3, E8, G8, H8	VDD_IO	Supply	Supply for digital IOs: 2.8 V nominal.
H5	VDD_DAC	Supply	Supply for video DAC: 2.8 V nominal.
A8	VDD_PLL	Supply	Supply for PLL: 2.8 V nominal.
B4, H6	VAA	Supply	Analog power: 2.8 V nominal.
H7	VAA_PIX	Supply	Analog pixel array power: 2.8 V nominal. Must be at same voltage potential as VAA.
H4	Reserved		Must be left floating for normal operation.
B5, C5, D1, D5, H1	DGND	Supply	Digital ground.
E5, F4	AGND	Supply	Analog ground.

If the VDAC is not used, that is, only parallel output is used, DAC_REF, DAC_POS, and DAC_NEG can be left floating. However, VDD_DAC must be tied to 2.8 V.

If the host config mode is required, SPI_SDI pin must be tied to GND.

DAC Video Output Design

The ASX340AT has a 10−bit current steering DAC designed for video signaling. An external resistor RREF for DAC_REF determines the full scale output signal current of the DAC. The output currents are converted to an output voltage by external resistors for DAC_POS and DAC_NEG. Table 5 shows the current of the DAC with different DAC_REF values.

DAC video output can be configured in either a differential mode or in a single−ended mode. To qualify for the output voltage specification, the following circuit designs are recommended:

- For single−ended mode, DAC_NEG is required to be properly terminated to GND. Figure 4 shows the recommendations when DAC_POS is connected to a passive low pass filter.

Table 5. SETTING FULL SCALE SIGNAL OUTPUT CURRENT

R_{REF}(W)	I_{FS}(mA)
2400	37.333
3200	28
4800	18.666

NOTE: Capacitance on the reference resistor must be 5pF or less to maintain reference stability

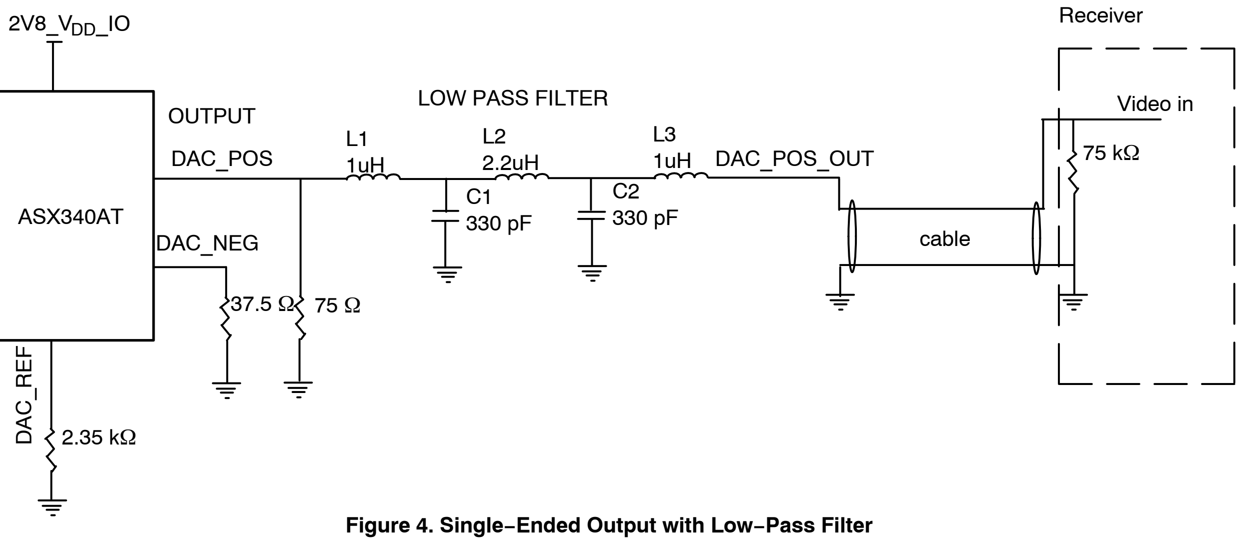


Figure 4. Single−Ended Output with Low−Pass Filter

Figure 6 shows the recommendations when DAC_POS is connected to a high input impedance component.

Each of the recommended single−ended DAC configuration’s outputs vary between 0 and 1.4 V. To change the output voltage level, adjust R_{REF} value within the limits of 2400 Ω and 4800 Ω.

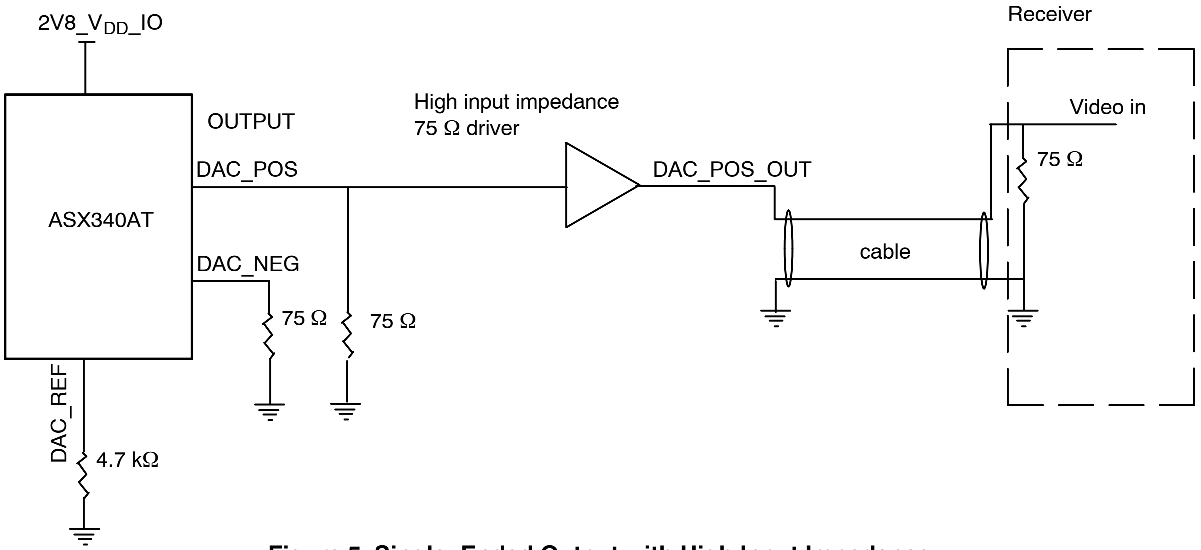


Figure 5. Single−Ended Output with High Input Impedance

For differential output, DAC_REF is recommended to connect to 2.35 kΩ to GND, and DAC_POS and DAC_NEG are both recommended to be grounded via a 75 Ω resistor. See Figure 6 for example.

The DAC output voltage of this recommended differential configuration is between −1.4 V and 1.4 V. To change the output voltage level, adjust R_{REF} value within the limits of 2400 Ω and 4800 Ω.

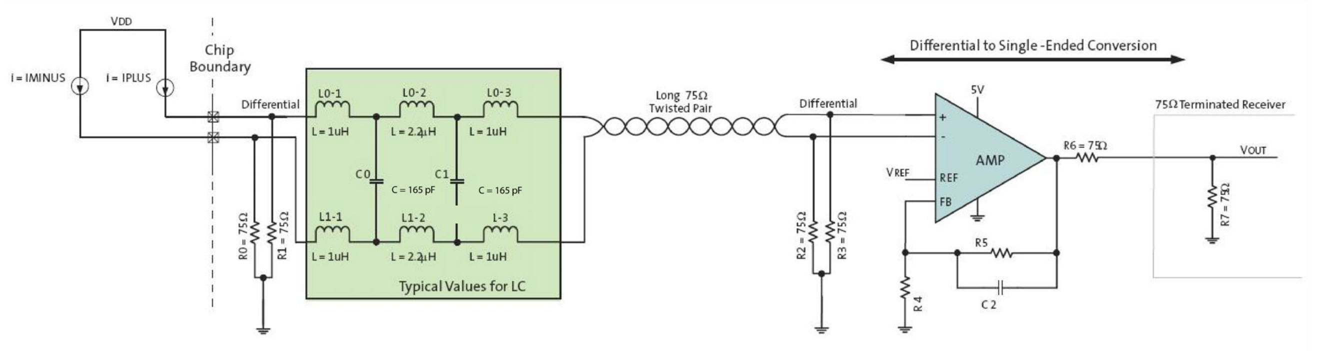


Figure 6. Differential−Ended Circuit Example

V_{DD} Internal Regulator Usage

The ASX340AT includes an internal voltage regulator that can be used, in conjunction with a suitable external transistor, to generate the core logic 1.8 V VDD power supply from the analog 2.8 V VAA supply. It should only be considered for VDD power supply usage. It is intended to replace the external low drop−out (LDO) regulator that is normally employed for this purpose as an option to reduce the cost of the bill−of−materials, at the expense of an extra package pin and additional production test steps to ensure that the regulator works. It does not provide superior performance and does not include thermal or current limiting, but is able to satisfy the current required by the ASX340 for operation of the device within its specification limits. In cases in which the modest cost savings between an LDO and a plain transistor are not significant, the LDO is recommended.

To utilize this internal circuit to form a voltage regulator, it needs to be attached to an off−chip bipolar junction transistor (BJT) and reservoir capacitor as shown in Figure 7. The regulator was designed to use a BSR16 transistor, but the choice is not restricted to this device alone. To ensure a stable loop for the regulator to work reliably, requirements for choosing the transistor and capacitor are tabled below. With meeting these requirements, the sensor is expected to function properly with the internally generated 1.8 V supply.

Table 6. EXTERNAL BIPOLAR TRANSISTOR PARAMETERS

Parameter	Symbol	Min	Typical	Max	Units
Current gain	h_{fe}	75		300	A/A
Transition frequency	f_T	200			MHz
Collector current	$I_{c(max)}$	400			mA

Table 7. EXTERNAL RESERVOIR CAPACITOR PARAMETERS

Parameter	Symbol	Min	Typical	Max	Units
Capacitance	C_{dec}	3.3	4.7		μF
Equivalent series resistance	R_{esr}	10	30	100	mΩ

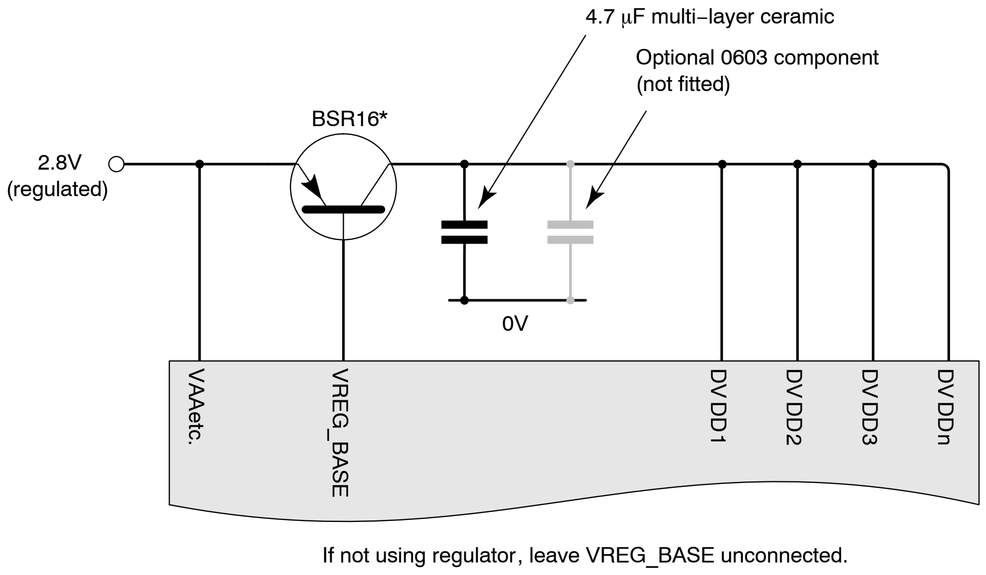

4.7 μF multi−layer ceramic
Optional 0603 component (not fitted)
BSR16*
2.8V (regulated)
0V
VAAetc.
VREG_BASE
DVDD1
DVDD2
DVDD3
DVDDn
If not using regulator, leave VREG_BASE unconnected.

Figure 7. Block Diagram Using VDD Internal Regulator

INITIALIZATION

This section explains the initialization of the sensor system including the power−up sequence, boot−up mode configuration, boot−up issues and solutions, and so on.

Power Up and Power Down Sequence

Please refer to the data sheet for details on power up and power down sequence.

Hard Reset

A hard reset is asserted or de−asserted with the RESET_BAR pin, which is active LOW. A hard reset can also be triggered by setting bit 1 of the register RESET_AND_MISC_CONTROL on SYSCTL page. In the reset state, all control registers are set to default values. The output states after hard reset are shown in Table 8. The Notes column specifies the properties of each pin, as well as how these pins should be connected during normal operation and when not used. For the pins that can be left unconnected/floating, it is explicitly specified so in the Notes column.

The ASX340AT also includes a power−on reset (POR) feature that initiates a reset upon power−up of the ASX340AT.

Table 8. RESET/DEFAULT STATE OF INTERFACES

Name	Reset State	Default State	Notes
EXTCLK	Clock running or stopped	Clock running	Input
XTAL	N/A	N/A	Input
RESET_BAR	Asserted	De−asserted	Input
SCLK	N/A	N/A	Input. Must always be driven to HIGH via a pull−up resistor in the range of 1.5 to 4.7 kW.
SDATA	High impedance	High impedance	Input/Output. Must always be driven to high via a pull−up resistor in the range of 1.5 to 4.7 kΩ.
SADDR	N/A	N/A	Input. Must be permanently tied to VDD_IO or GND.
SPI_SCLK	High impedance	Driven, logic 0	Output. Output enable is R0x0032[13].
SPI_SDI	Internal pull−up enabled	Internal pull−up enabled	Input. Internal pull−up is permanently enabled.
SPI_SDO	High impedance	Driven, logic 0	Output enable is R0x0032[13].
SPI_CS_N	High impedance	Driven, logic 1	Output enable is R0x0032[13].
FRAME_VALID	High impedance	High impedance	nput/Output. This interface is disabled by default. Input buffers (used for GPIO function) powered down by default, so these pins can be left unconnected (floating). After reset, these pins are powered up, sampled, then powered down again as part of the auto−configuration mechanism. See Note 9.
LINE_VALID			
PIXCLK	High impedance	Driven, logic 0	Output. This interface disabled by default. See Note 8.
DOUT7			
DOUT6			
DOUT5			
DOUT4			
DOUT3			
DOUT2			
DOUT1			
DOUT0			
DOUT_LSB1	High impedance	High impedance	Input/Output. This interface disabled by default. Input buffers (used for GPIO function) powered down by default, so these pins can be left unconnected (floating). After reset, these pins are powered−up, sampled, then powered down again as part of the auto−configuration mechanism.
DOUT_LSB0	High impedance	High impedance	
DAC_POS	High impedance	Driven	Output. Interface disabled by hardware reset and enabled by default when the device starts streaming.
DAC_NEG			
DAC_REF			

Table 8. RESET/DEFAULT STATE OF INTERFACES (continued)

Name	Reset State	Default State	Notes
TDI	Internal pull−up enabled	Internal pull−up enabled	Input. Internal pull−up means that this pin can be left unconnected (floating).
TDO	High impedance	High impedance	Output. Driven only during appropriate parts of the JTAG shifter sequence.
TMS	Internal pull−up enabled	Internal pull−up enabled	Input. Internal pull−up means that this pin can be left unconnected (floating).
TCK	Internal pull−up enabled	Internal pull−up enabled	Input. Internal pull−up means that this pin can be left unconnected (floating).
TRST_N	N/A	N/A	Input. Must always be driven to a valid logic level. Must be driven to GND for normal operation.
FRAME_SYNC	N/A	N/A	Input. Must always be driven to a valid logic level. Must be driven to GND if not used.
GPIO12	High impedance	High impedance	Input/Output. This interface disabled by default. Input buffers (used for GPIO function) powered down by default, so these pins can be left unconnected (floating).
GPIO13	High impedance	High impedance	Input/Output. This interface disabled by default. Input buffers (used for GPIO function) powered down by default, so these pins can be left unconnected (floating).
ATEST1	N/A	N/A	Must be driven to GND for normal operation
ATEST2	N/A	N/A	Must be driven to GND for normal operation

8. The reason for defining the default state as logic 0 rather than high impedance is this: when wired in a system (for example, on **onsemi**'s demo boards), these outputs will be connected, and the inputs to which they are connected will want to see a valid logic level. No current drain should result from driving these to a valid logic level (unless there is a pull−up at the system level).
9. These pads have their input circuitry powered down, but they are not output−enabled. Therefore, they can be left floating but they will not drive a valid logic level to an attached device.

Soft Reset

A soft reset has the same effect as a hard reset. In soft reset mode, the two−wire serial interface and the register bus are still running. Soft reset is asserted or de−asserted by setting bit 0 of the register RESET_AND_MISC_CONTROL on SYSCTL page.

Soft Standby

There is no hard standby mode supported by ASX340AT; however, the system can enter a soft standby mode by issuing host command interface (HCI) commands:

```
//Entering standby mode:
REG= 0xFC00, 0x5000      // CMD_HANDLER_PARAMS_POOL_0
REG= 0x0040, 0x8100      // COMMAND_REGISTER
POLL_FIELD= COMMAND_REGISTER, DOORBELL, !=0, DELAY=10, TIMEOUT=100
ERROR_IF= COMMAND_REGISTER, HOST_COMMAND, !=0, "Command failed"
//Exiting standby mode:
REG= 0xFC00, 0x5400      // CMD_HANDLER_PARAMS_POOL_0
REG= 0x0040, 0x8100      // COMMAND_REGISTER
POLL_FIELD= COMMAND_REGISTER, DOORBELL, !=0, DELAY=10, TIMEOUT=100
ERROR_IF= COMMAND_REGISTER, HOST_COMMAND, !=0, "Command failed"
```

In soft standby mode, the total power consumption is around 24 mW (for reference only), which is about 10% of the total power consumed when using the composite output only. In this mode, the whole device is switched off except the two−wire serial interface and parts of the system control that control the clocks and interrupts. A further power reduction can be achieved by turning off the input clock, but this must be restored before issuing the host commands to restart the device.

System Configuration and Usage Modes

How a camera based on the ASX340AT will be configured depends on what features are used. There are essentially three configuration modes for ASX340AT:

- Auto−Config Mode
- Flash Config Mode
- Host Config Mode

Auto−Config Mode

In the simplest case, an ASX340AT operating in Auto−Config mode with no customized settings might be sufficient. As shown in Figure 8, this is truly a single chip operation with no EEPROM/Flash or microcontroller.

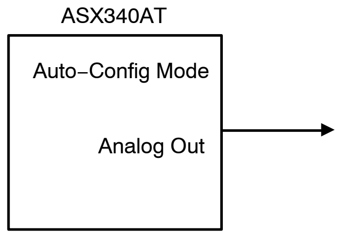


Figure 8. Auto−Config Mode

Flash Config Mode

The ASX340AT can be configured by a serial EEPROM or Flash through the SPI Interface as shown in Figure 9. Flash image sizes vary depending on the data for registers, firmware, and overlay data. Overlay data can alternatively be issued by the external microcontroller if the rate of refreshing data is deemed adequate.

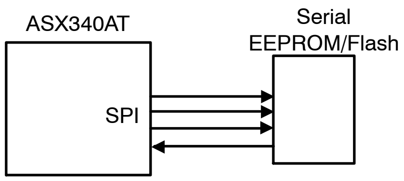


Figure 9. Flash Config Mode

Functions such as overlay or 2x zoom can also be assigned to general purpose inputs, as shown in Figure 10. That capability can be employed on all configurations with external EEPROM or Flash memory by mapping overlay images to an input. This is achieved through SET_GPI_ASSOCIATION and command sequences. Refer to Command Sequence section for details.

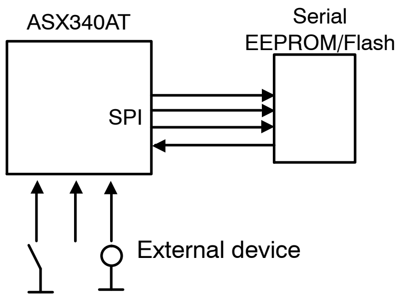


Figure 10. Flash Config Mode

Host Config Mode

In host−config mode, the ASX340AT is configured with settings directly passed by a host micro controller. A back−up camera with dynamic input from the steering system will require a microcontroller with a system bus interface. Typically, a system bus can be connected to a rear−view camera for the purpose of dynamically providing steering information that will in turn be translated into overlay images being loaded and displayed by the microcontroller as shown in Figure 11.

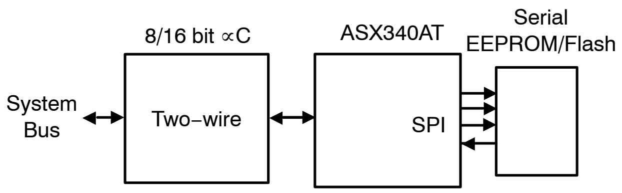


Figure 11. Host Config Mode with Flash

Overlay information may also be passed by the microcontroller without a need for an EEPROM or Flash memory as shown in Figure 12. However, because the data transfer rate is limited over the two−wire serial bus, the update rate may be slower. However, overlay images can be preloaded into the four on−chip buffers, in which case they may be turned on and off or move locations at the frame rate.

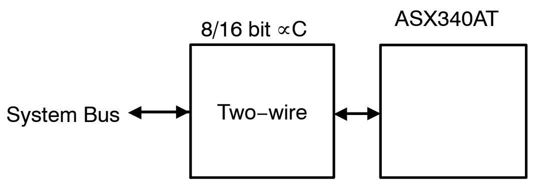


Figure 12. Host Config Mode

Boot−up Mode Configuration

After power is applied and the device is out of reset by de−asserting the RESET_BAR pin, it will enter a boot sequence to configure its operating mode. The ASX340AT firmware supports a System Configuration phase at start−up, entered immediately after the ASX340AT power−up or reset. This consists of five steps of execution in sequence (see Figure 13).

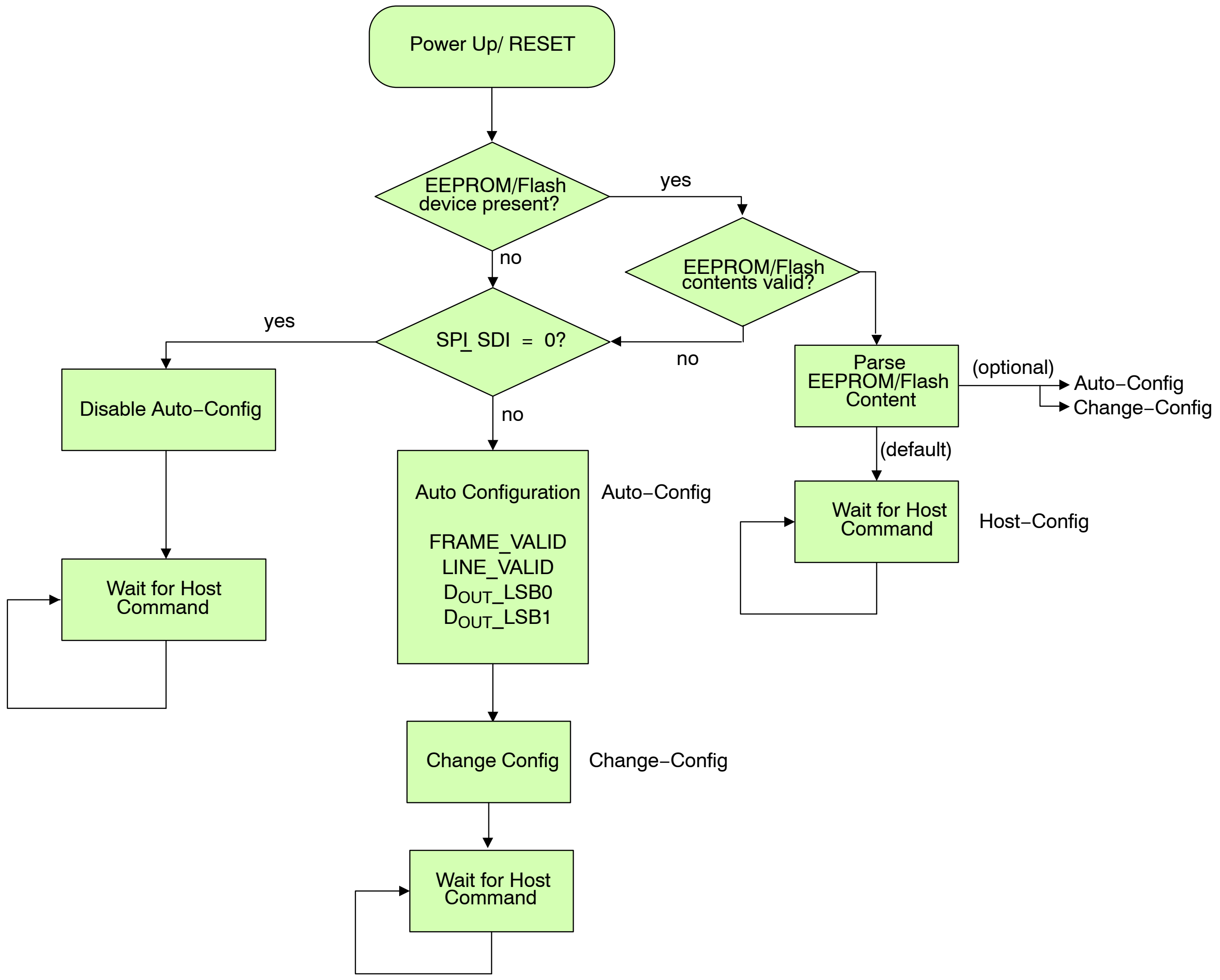


Figure 13. Power−up Sequence − Configuration Options Flow Chart

2. Flash Detection
   It first enters the Flash Detection mode, which attempts to detect the presence of an SPI Flash or EEPROM device:
   If no device is detected, the firmware then samples the SPI_SDI pin state to determine the next mode:
   - If SPI_SDI = 0 then it enters the Host−Config mode (Step 4).
   - If SPI_SDI = 1 then it enters the Auto−Config mode (Step 3).

   If a device is detected, the firmware switches to the Flash−Config mode (Step 2).
3. Flash−Config
   In the Flash−Config phase, the firmware interrogates the device to determine if it contains valid configuration records:
   - If no records are detected, then the firmware enters the Auto−Config mode.
   - If records are detected, the firmware processes them. By default, when all Flash records are processed the firmware switches to the Host−Config mode. However, the records encoded into the Flash can optionally be used to instruct the firmware to proceed to one of the other modes (auto−config/change−config).
4. Auto−Config
   The Auto−Config mode uses the FRAME_VALID, LINE_VALID, DOUT_LSB0 and DOUT_LSB1 pins to configure the operation of the device, such as video format and pedestal (see Table 9). After Auto−Config completes, the firmware switches to the Change−Config mode. The auto−config mode is only available for analog output.

Table 9. GPIO BIT DESCRIPTIONS

	GPIO[11] (D_{OUT}_LSB1)	GPIO[10] (D_{OUT}_LSB0)	GPIO[9] (FRAME_VALID)	GPIO[8] (LINE_VALID)
Low (“0”)	Normal	NTSC	Normal	No pedestal
High (“1”)	Vertical Flip	PAL	Horizontal mirror	Pedestal

5. Host−Config
   In the Host−Config mode, the firmware performs no configuration, and remains idle waiting for configuration and commands from the host. The System Configuration phase is effectively complete and the SOC will take no actions until the host issues commands.
6. Change−Config (commences streaming − completes the System Configuration mode).
   In the Change−Config mode, the firmware performs a ‘Change−Config’ operation. This applies the current configuration settings to the SOC, and commences streaming. This completes the System Configuration phase.

Boot−up Issues and Solutions

- Failed to boot up in the flash config mode:
  This is mostly due to corrupted bin files. There are two possible failures:
  - Boots up in auto−config mode instead. In this case, just reprogram the flash memory using a different bin file and try to boot up in the flash config mode again.
  - There is no output. In this case, there are two ways to recover:
    1. First disconnect SPI_SDI pin, and boot up in auto−config mode. Then connect SPI_SDI pin back to SDO of flash memory while power is on. Now program the flash memory again with a working bin file.
    2. First set spi_config_disable, i.e. R0x0020[5] = 1 and reset_soft, i.e. R0x001A[0] = 1; then release reset_soft, i.e. R0x001A[0] = 0, and clear spi_config_disable, i.e. R0x0020[5] = 0. Now program the flash memory with a working bin file.

Device ID

The ASX340AT provides device ID for identifying the device after power−up by the host processor.

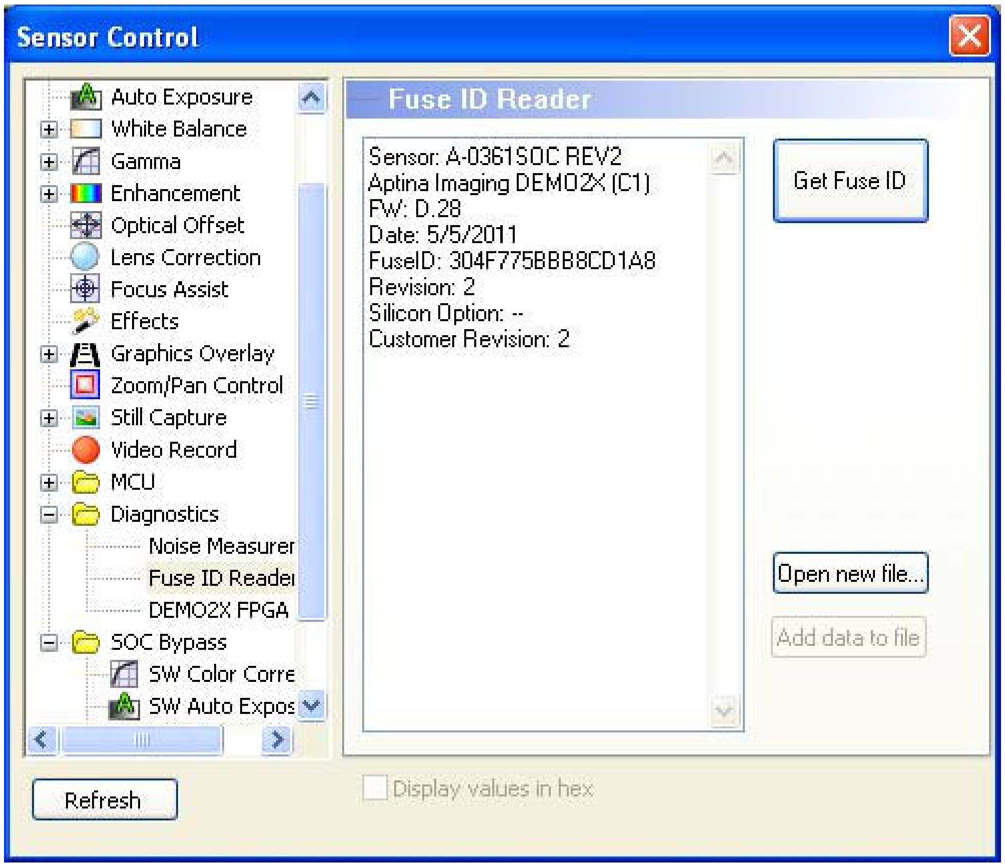
Table 10. DEVICE ID RELATED REGISTERS AND VARIABLES

Map	Address	Bits	Description
SYSCTL	0X0000	[15:0]	Contains the ASX340AT device ID Number, 0x2285. Read−only.

Fuse ID

Fuse ID for the ASX340AT can also be obtained through DevWare. Fuse ID is helpful in determining the source of the sensors, that is, the wafer, lot, and so on, to trace back on the history of the sensors. It is 64−bit fuse ID data saved in four 16−bit registers, i.e. FUSE_ID1, FUSE_ID2, FUSE_ID3, and FUSE_ID4 (R0x31F4, R0x31F6, R0x31F8, and R0x31FA).

Go to *Control* → *Diagnostics* → *Fuse ID Reader*, and click on *Get Fuse ID*. The sensor information will then be populated into the box as shown in Figure 14.



Sensor: A-0361SOC REV2
Aptina Imaging DEMO2X (C1)
FW: D.28
Date: 5/5/2011
FuseID: 304F775BBB8CD1A8
Revision: 2
Silicon Option: --
Customer Revision: 2

Figure 14. Fuse ID Reader in DevWare

PROGRAMMING REGISTERS AND VARIABLES

This developer guide refers to various memory locations and registers that the user reads from or writes to for altering the ASX340AT operation. Hardware registers may be read or written by sending the address and data information over the two−wire serial interface.

Accessing Registers and Variables

The host can control the ASX340AT in three ways:

- By issuing commands to the embedded microcontroller
- By reading and writing firmware variables, which influence the operation of the embedded microcontroller
- By reading and writing hardware registers

In each case, the physical interface to the ASX340AT is the two−wire serial interface, using 16−bit addresses. Where possible, the ASX340AT should be controlled through commands and variables since these have been designed to provide correctly−sequenced control of the underlying hardware. In contrast, access to registers is discouraged, since it may cause undesired interaction with microcontroller operations. Do not change any of the reserved bits.

A summary of all the available registers and variables are listed in Table 11. The detailed explanations of these registers and variables can be found in the Register and Variable Reference document.

Table 11. SUMMARY OF REGISTERS AND VARIABLES

Registers

	Adrress	Map	Functions
	0x3002−0x31FA	Core	core
	0x3640−0x3784	SOC2	IFP
	0x0000−0x0040	SYSCTL	Clocks, Slew control
	0x0982−0x099E	XDMA	RAM address pointer
	0x3C08−0x3C28	TX_SS	Parallel

Variables

Driver Number	Offset Address	Direct XDMA Address Range	Map
0	0x00−0x06	0x8000−0x800E	Monitor
5	0x20−0x2C	0x9420−0x942C	NTSC
6	0x20−0x2C	0x9820−0x982C	PAL
9	0x04−0x20	0xA404−0xA420	AE_Rule
10	0x00−0x57	0xA800−0xA857	AE_Track
11	0x00−0x16	0xAC00−0xAC16	AWB
12	0x04−0x0D	0xB004−0xB00D	BlackLevel
13	0x02−0x2B	0xB402−0xB42B	CCM
15	0x02−0x3C	0xBC02−0xBC3C	Low Light
16	0x00−0x80	0xC000−0xC080	Flicker Detect
18	0x00−0x18A	0xC800−0xC98A	CamControl
23	0x00−0x11	0xDC00−0xDC11	System Mgr
24	0x00−0x1C	0xE000−0xE01C	Patch Loader
31	0x00−0x0E	0xFC00−0xFC0E	Command Handler

Registers

Registers can be accessed by the two−wire serial interface with addresses in the range 0x0000−0x7FFE. All registers are 16−bits in size and register access only supports 16−bit data read and write. Figure 15 shows how the register is referenced. For example, to write 0x020B to y_addr_end (direct address is 0x3006):
REG = 0x3006, 0x020B

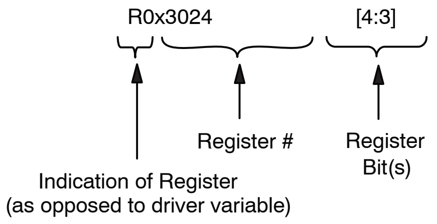


Figure 15. Register legend

Variables

Variables correspond to locations in the memory space of the embedded microcontroller. Variables can be accessed by the two−wire serial interface with addresses in the range 0x8000−0xFFFF. Variables can be 8, 16 or 32−bit in size and variable access supports access of any 8−bit multiple.

Variables are divided into groups called *Drivers*. Each variable is specified by a driver number (0...31) and an offset. Figure 16 shows the legend of variables. This document uses the notation VAR (driver_number, offset). Given a driver number and offset, the corresponding direct address is calculated like this:

Direct−Address = 0x8000 | (driver_number << 10) | offset

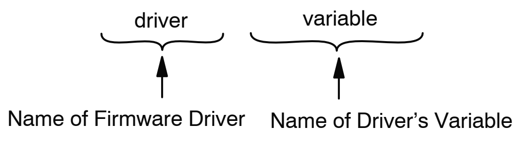


Figure 16. Firmware Variable Legend

For example, ae_rule_algo is VAR(0x09, 0x0004). Its direct address is therefore 0x8000 |(9<<10) | 4 = 0xA404.

To access variables when using DevWare INI files, ASX340AT supports two approaches:

- VAR = driver_number_1, offset_1, value_1
- VAR = driver_number_2, offset_2, value_2
  …… and
- REG = direct_address_1, value_1
- REG = direct_address_2, value_2
- ……

For example, to write 0x0002 to ae_rule_algo, i.e. VAR(0x09, 0x0004), the following two approaches are equivalent:

- Approach 1:
  VAR = 0x09, 0x0004, 0x0002
- Approach 2:
  REG = 0xA404, 0x0002

Change Config Command

When some register values are changed, for example variables for mirror and flip, a change−config command must be issued before the new settings will take effect. Refer to the Register and Variable Reference document to determine which other registers need this command.

- When issuing the *change config* command for records stored in and executed from NVM (for example, Flash or EEPROM), the ***change config*** command has to be issued depending on which section it is required:
  - If a *change config* command is required during the system configuration phase, e.g. in the Init table sections of an fcfg file, it has to be issued by setting SYSMGR_CONFIG_MODE, that is, VAR(0x12, 0x000C) to 0x04:
    VAR = 0x12, 0x000C, 0x04

    When converting it to fcfg command using FlashTool, it is written as follows:
    ```
    TYPE=create_var_set_v1
    PARAMETERS={
    0x17,0xC,8,0x4 # SYSMGR_CONFIG_MODE
    }
    ```
    Only one change−config command is needed in the entire Init table because the change−config command is executed on completion of the System Configuration phase.
  - If a change−config command is required in a Command Sequence, it has to be issued by a host command – Set_State:
    Set_State, 0x28
    There is one complication: if an Init Table contains the MISC_INVOKE_COMMAND_SEQ command, to invoke a command sequence during System Configuration phase, this command sequence cannot contain the SET_STATE command. It must use the SYSMGR_CONFIG_MODE variable for a change−config request. Refer to the Host Command Interface document for more details on Change−Config command usage in flash records.
- When in host mode, through the serial interface, for example, the *change config* command has to be issued through HCI commands:
  REG= 0xFC00, 0x2800//CMD_HANDLER_PARAMS_POOL_0
  REG= 0x0040, 0x8100//COMMAND_REGISTER
  POLL_FIELD= COMMAND_REGISTER, DOORBELL, !=0, DELAY=10, TIMEOUT=100
  ERROR_IF= COMMAND_REGISTER, HOST_COMMAND, !=0, ”Command failed”

Host Command Interface

The ASX340AT supports a host command interface, which allows writes to registers and variables. The host issues a 16−bit command to the device by performing a register write to the command register (SYSCTL 0x40). More details can be found in “Host Command Interface” as well as the ASX340AT Host Command Interface document.

BASIC CONFIGURATION

Clock and PLL Control

The ASX340AT has two primary clocks:

- A master clock coming from the EXTCLK signal. To generate NTSC or PAL format images, the sensor core requires a 27 MHz clock for EXTCLK.
- In default mode, a pixel clock (PIXCLK) running at 2 * EXTCLK. In raw Bayer bypass mode, PIXCLK runs at the same frequency as EXTCLK.

When the ASX340AT operates in sensor stand−alone mode, the image flow pipeline clocks can be shut off to conserve power. The sensor core is a master in the system. The sensor core frame rate defines the overall image flow pipeline frame rate. Horizontal blanking and vertical blanking are influenced by the sensor configuration, and are also a function of certain image flow pipeline functions. The relationship of the primary clocks is depicted in Figure 17. The image flow pipeline typically generates up to 16 bits per pixel−for example, YCbCr or 565RGB−but has only an 8−bit port through which to communicate this pixel data.

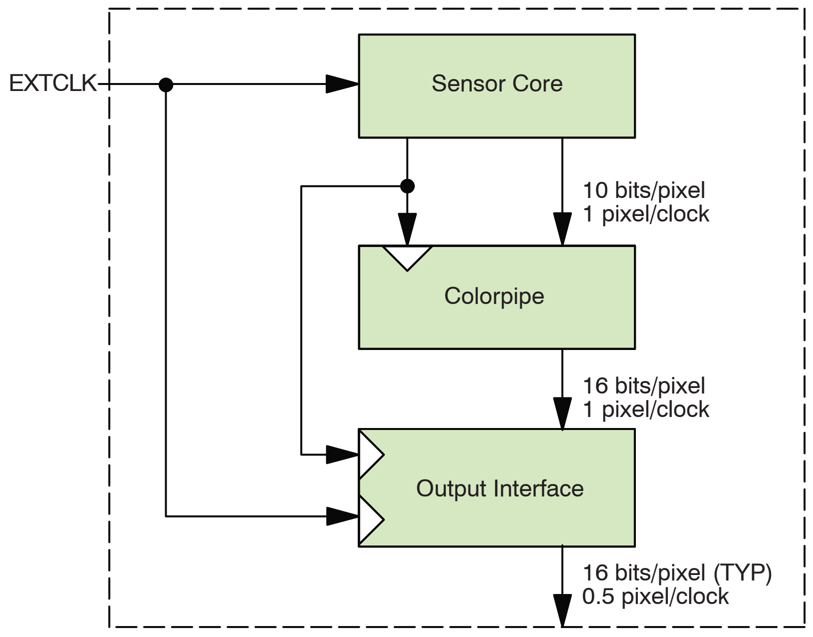


Figure 17. Primary Clock Relationship

Slew Rate Control

System power consumption, noise, and electromagnetic interference (EMI) are reduced by selecting the optimum slew rate to meet the timing budget.

Figure 18 for example shows how slew rate is measured on PIXCLK and DOUT. The ASX340AT output slew rate control register is pad_slew (R0x001E on the SYSCTL page). Details are shown in Table 12.

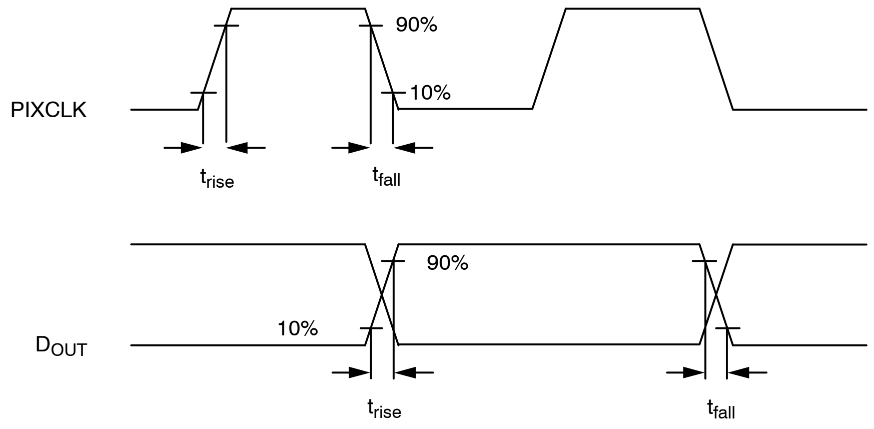


Figure 18. Slew Rate Timing

Table 12. SLEW RATE CONTROL REGISTERS – R0x001E

Bits	Name	Description
[10:8]	slew_pixclk	Controls the slew rate of PIXCLK independent of other outputs
[6:4]	slew_spi	Controls the SPI bus slew rates
[2:0]	slew_dout	Controls the slew rate of DOUT[7:0], DOUT_LSB1, DOUT_LSB0, FRAME_VALID (FV), LINE_VALID (LV), GPIO13 and GPIO12.

SDATA and SCLK have no rise time slew rate control. SDATA has an open drain output without an active p−channel transistor. Slew rate control is accomplished by an external passive resistor.

Table 13 is the look−up table between slew rates and actual slew rate timing. Eight slew rates (code 0−code 7) are available. Code 0 is the slowest; code 7 is the fastest. Rise time and fall time are typically matched.

Table 13. SLEW RATE FOR PIXCLK AND D_{OUT}

f_{EXTCLK} = 27 MHz; V_{DD} = 1.8 V; VDD_IO = 2.8 V; VAA = 2.8 V; V_{AA}_PIX = 2.8 V; VDD_PLL = 2.8 V; VDD_DAC = 2.8 V; T = 25°C; CLOAD = 40 pF

PIXCLK			D_{OUT}[7:0]			Unit
R0x1E[10:8]	Rise Time	Fall time	R0x1E[2:0]	Rise time	Fall Time	
000	NA	NA	000	15.0	13.5	ns
001	NA	NA	001	9.0	8.5	ns
010	7.0	6.9	010	6.8	6.0	ns
011	5.2	5.0	011	5.2	4.8	ns
100	4.0	3.8	100	3.8	3.5	ns
101	3.0	2.8	101	3.3	3.3	ns
110	2.4	2.2	110	3.0	3.0	ns
111	1.9	1.7	111	2.8	2.8	ns

Figures 19 and 20 show the effect of different slew rates. The users should realize that the default slew rate values are not designed for a particular system; hence, all users are advised to optimize their settings based on their system designs. The following examples show different settings for slew rate and the effect that an incorrect setting of slew rate can have on the received image. Incorrect settings can lead to incorrect sampling of the data stream.

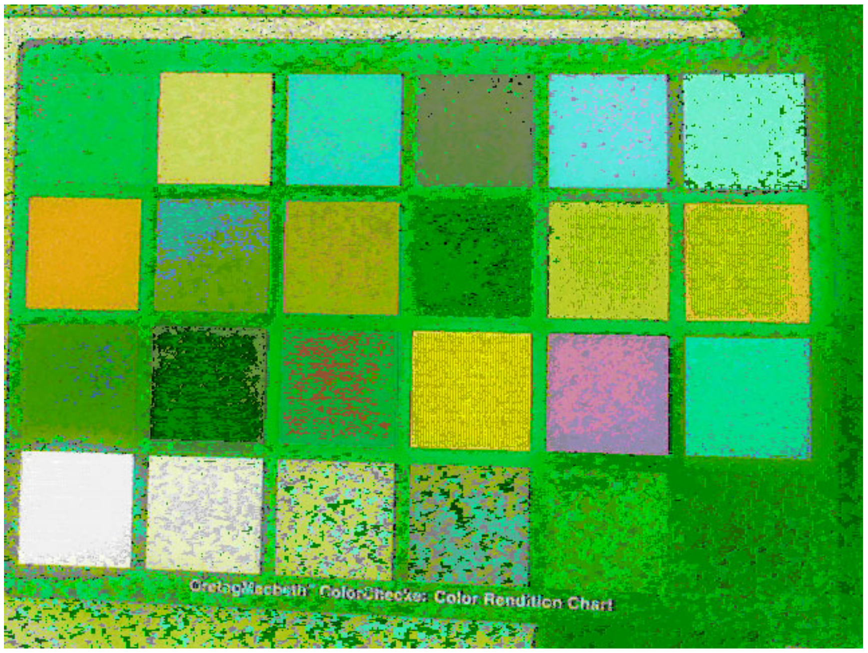


Figure 19. Slew_Dout – 0x04

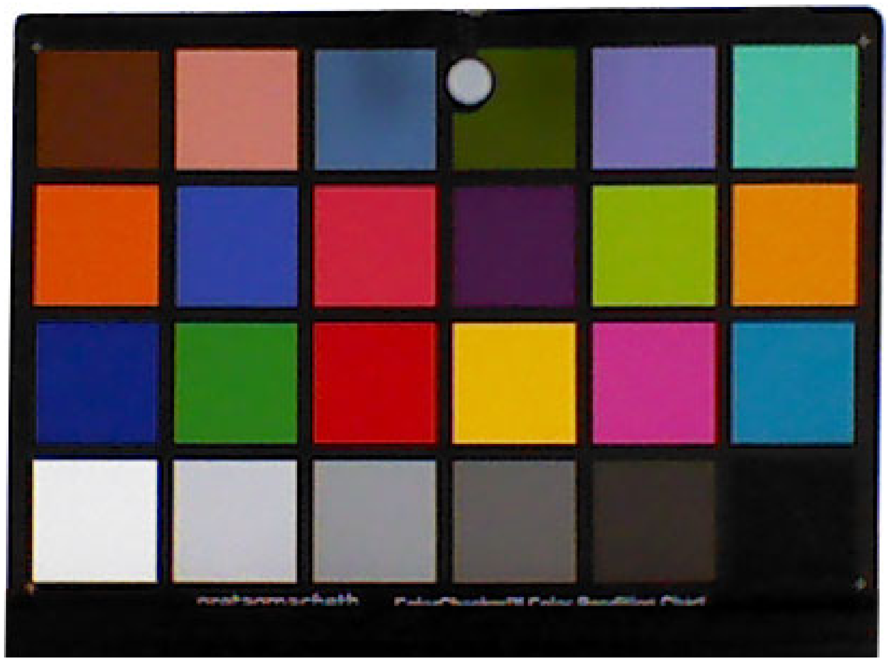


Figure 20. Slew_Dout – 0x00

Analog and Parallel Output

ASX340AT supports VGA progressive output with NTSC or PAL analog output enabled. To enable this, use the following presets. Do a change−config after each preset for it to take effect.

[NTSC: Enable VGA Progressive on Parallel Port]
REG = 0x9426, 0x25
[PAL: Enable VGA Progressive on Parallel Port]
REG = 0x9826, 0x25

Output Format

The ASX340AT supports the following pixel formats:

- 16−bit YCbCr (default)
- 16−bit 565RGB
- 15−bit 555RGB
- 12−bit 444xRGB
- 12−bit x444RGB
- (8 + 2)−bit Processed Bayer output
- 10−bit raw Bayer output

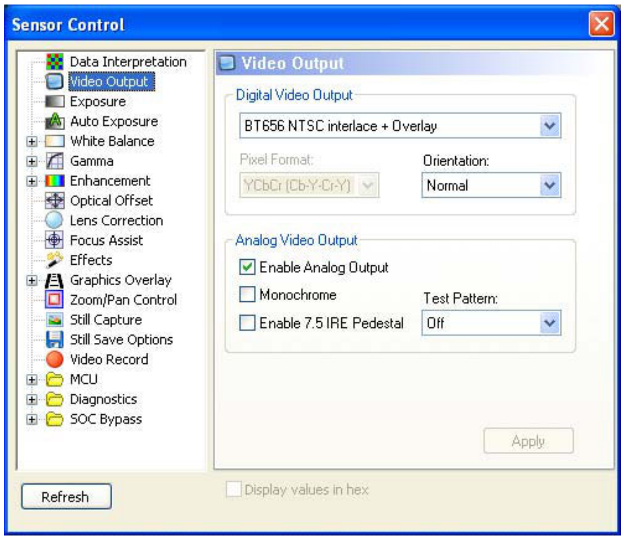
For specific data ordering of each pixel format, please refer to the data sheet of ASX340AT. The ASX340AT can output processed video as a standard ITU−R BT.656 (CCIR656) stream, an RGB stream, or as unprocessed Bayer data. The ASX340AT also supports NTSC and PAL formats in both analog and digital video output. There are also two scanning modes available: interlace and progressive:

- For interlace mode, both analog and digital video output are supported only in YCbCr;
- For progressive mode, only digital video output is supported, but in any of the pixel formats listed above. Raw 10−bit Bayer data is only available in progressive mode.

Since the data output signal is 8 bits wide, 12−, 15−, 16−, or (8+2)−bit data are output in a two−byte sequence. For raw 10−bit Bayer data, the two least significant bits are output on DOUT_LSB[1:0] signals.

For raw data output, ASX340AT supports a maximum output size of 648x488 in Bayer format. For NTSC mode, parallel output has 7 rows of black active image data, which can be observed in DevWare. Analog output will not show the dark rows. Due to the decoding of the BT656 codes, black columns can be observed in both NTSC and PAL digital output.

The output format can be configured through *Control →Video Output* in DevWare as shown in Figure 21:



Video Output
Digital Video Output: BT656 NTSC interlace + Overlay
Pixel Format: YCbCr (Cb-Y-Cr-Y)
Orientation: Normal
Analog Video Output
[✓] Enable Analog Output
[ ] Monochrome
[ ] Enable 7.5 IRE Pedestal
Test Pattern: Off

Figure 21. Configure Video Output in DevWare

Table 14 shows the variables and registers related to program the output data features. Refer to the ASX340AT Register and Variable Reference for more detailed descriptions for the registers and variables.

Table 14. REGISTERS AND VARIABLES TO PROGRAM VIDEO OUTPUT DATA

Map	Address	Name	Bits	Descriptions
CamControl	0xC858	cam_frame_scan_control	[4:3]	Progressive−scan control: 0: VGA60. 1: VGA50. 2: Custom. 3: Reserved.

Table 14. REGISTERS AND VARIABLES TO PROGRAM VIDEO OUTPUT DATA (continued)

Map	Address	Name	Bits	Descriptions
CamControl	0xC858	cam_frame_scan_control	[2:1]	Interlaced−scan control: 0: NTSC. 1: PAL. 2−3: Reserved.
CamControl	0xC858	cam_frame_scan_control	[0]	Scanning mode control: 0: Interlaced−scan. 1: Progressive−scan.
CamControl	0xC96C	cam_output_format	[13:12]	RGB output format: 0: 16−bit 565RGB. 1: 15−bit 555RGB. 2: 12−bit 444xRGB. 3: 12−bit x444RGB.
CamControl	0xC96C	cam_output_format	[11:10]	Select Bayer format: 0: Raw10. 3: Processed8+2.
CamControl	0xC96C	cam_output_format	[9:8]	Select output format: 0: YUV. 1: RGB. 2: Bayer. 3: None.
CamControl	0xC96C	cam_output_format	[2]	Enable monochrome (black and white) output: 0: Color. 1: Monochrome.
CamControl	0xC96C	cam_output_format	[1]	Swap output pixel high byte with low byte: 0: Don’t swap. 1: Swap.
CamControl	0xC96C	cam_output_format	[0]	Swap Red/Blue or Cr/Cb channels: 0: Don’t swap. 1: Swap.
CamControl	0xC972	cam_port_parallel_control	[4]	Read−only for interlaced mode; modifiable for progressive mode. Controls the pixel clock gating: 0: The pixel clock output (PIXCLK) is continuous. 1: The pixel clock output (PIXCLK) is only generated when FRAME_VALID and LINE_VALID are asserted.
CamControl	0xC972	cam_port_parallel_control	[2:1]	Read−only for interlaced mode; modifiable for progressive mode. Selects the parallel output source: 0: Reserved. 1: Interlaced. 2: Progressive. 3: Reserved.
CamControl	0xC972	cam_port_parallel_control	[0]	Read−only for interlaced mode; modifiable for progressive mode. Enables the parallel port: 0: Port disabled. 1: Port enabled.

Pedestal Configuration

ASX340AT supports 7.5 IRE pedestal to conform to North America's NTSC standard. When pedestal is enabled, the black level is raised by 7.5 IRE, which essentially reduces the dynamic range of the sensor. The pedestal, i.e. 7.5 IRE will be present in active video, but not in line blanking period.

Pedestal for NTSC is enabled by setting Bit 2 of NTSC_INTERLACED_PORT_COMPOSITE_CONTRO L (R0x9427[2]).

Reference information can be found at: http://www.itu.int/rec/R−REC−BT.470−6−199811−S/en

FOV Calibration and Stretch

CAM_FOV_CALIB_X_OFFSET and CAM_FOV_ CALIB_Y_OFFSET are FOV calibration variables.

The ASX340AT also supports stretch ability. Due to some analog display limitations, some analog output is not showing all the available pixels as shown in the digital output. By default, CAM_FOV_STRETCH_ACTIVE_PI XELS is 720, and CAM_FOV_STRETCH_FIRST_PIXEL is 0. By configuring these two variables, the analog output FOV will be adjusted. The digital output, however, will show black pixels when there is a left or right margin. The hardware supports a maximum of 14 pixels width for both margins. The left margin is defined by CAM_FOV_STRE TCH_FIRST_PIXEL. The right margin width is calculated as 720 − CAM_FOV_STRETCH_ACTIVE_PIXELS − CAM_FOV_STRETCH_FIRST_PIXEL, which guarantee s a 720 active pixel input each line to the TV encoder. Refer to Figure 22 for the illustration.

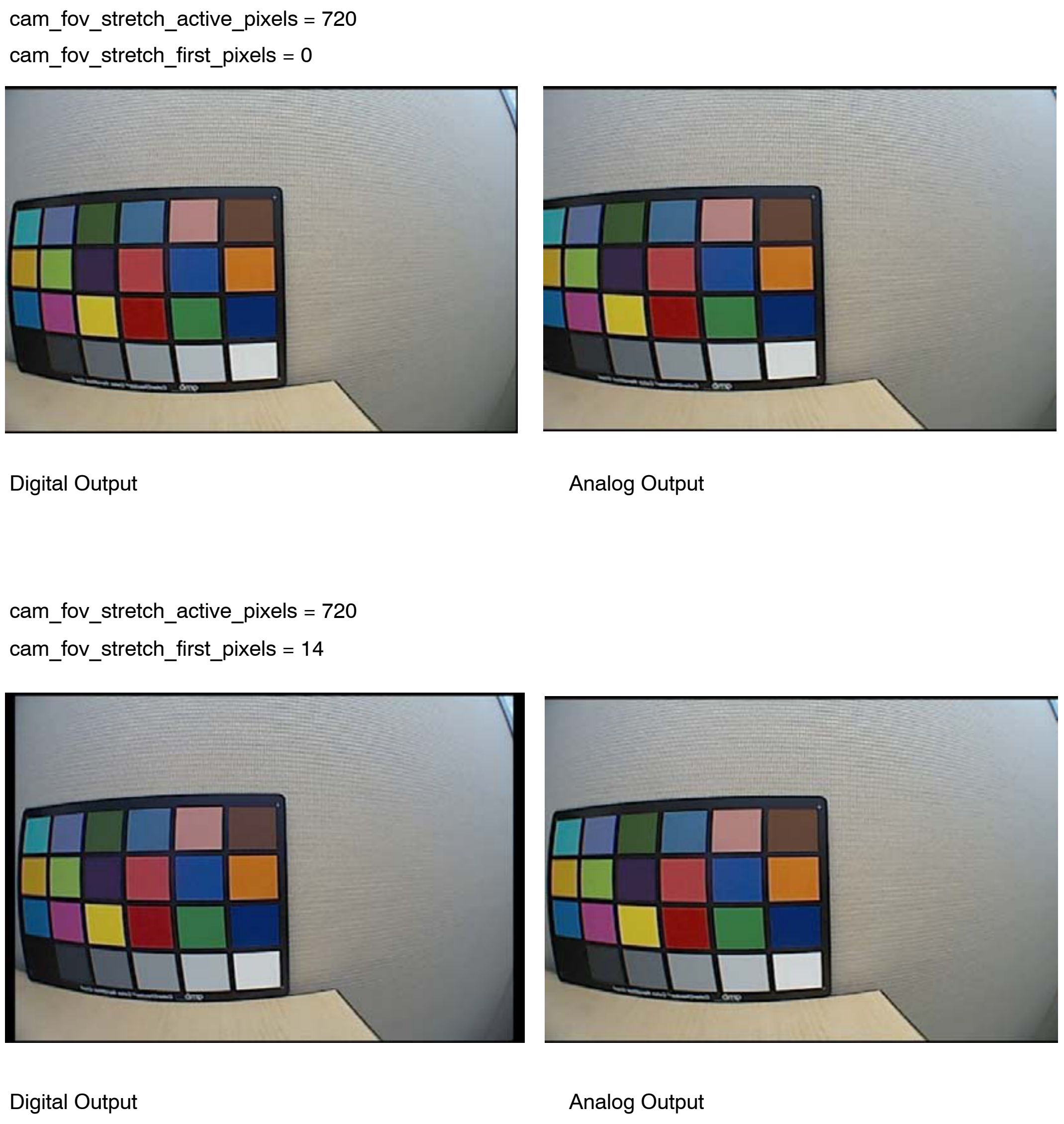

cam_fov_stretch_active_pixels = 720
cam_fov_stretch_first_pixels = 0
Digital Output
Analog Output
cam_fov_stretch_active_pixels = 720
cam_fov_stretch_first_pixels = 14
Digital Output
Analog Output

Figure 22. FOV Stretch

Table 15 summarizes the variables to adjust FOV calibration and stretch.

Table 15. FUNCTION TABLE

Map	Address	Name	bits	Descriptions
CamControl	0xC85C	cam_fov_calib_x_offset	[7:0]	Horizontal calibration offset for the sensor array. The min and max values are −41 to 41 in the NTSC/PAL case. This value is signed 2’s complement.
CamControl	0xC85D	cam_fov_calib_y_offset	[7:0]	Vertical calibration offset for the sensor array. The min and max values are −36 to 36 in the NTSC/PAL case. This value is signed 2’s complement.
CamControl	0xC85E	cam_fov_stretch_active_pixels	[15:0]	Width of the active line in pixels. Has to be an even number between 692 and 720. Inactive pixels will be black.
CamControl	0xC860	cam_fov_stretch_first_pixel	[7:0]	Width of the left margin. Has to be an even number smaller than or equal to 14.

10. All the changes take effect with Change−Config command.

Image Orientation

Image orientation can be configured through *Control→ Video Output* in DevWare as shown in Figure 21:

The variable that controls image orientation is CAM_SENSOR_CONTROL_READ_MODE [1:0] (R0x C838[1:0]):

- cam_sensor_control_read_mode[0], i.e. cam_sensor_control_horz_mirror_en, controls the horizontal mirror function
- cam_sensor_control_read_mode[1], i.e. cam_sensor_control_vert_flip_en, controls the vertical flip function.

Table 16 summarizes how image configuration is configured.

Table 16. IMAGE ORIENTATION CONTROL

cam_sensor_control_ read_mode[1:0]	Descriptions
0	Normal
1	Horizontal Mirror
2	Vertical Flip
3	Rotate 1805

NOTE: Needs change−config to take effect.

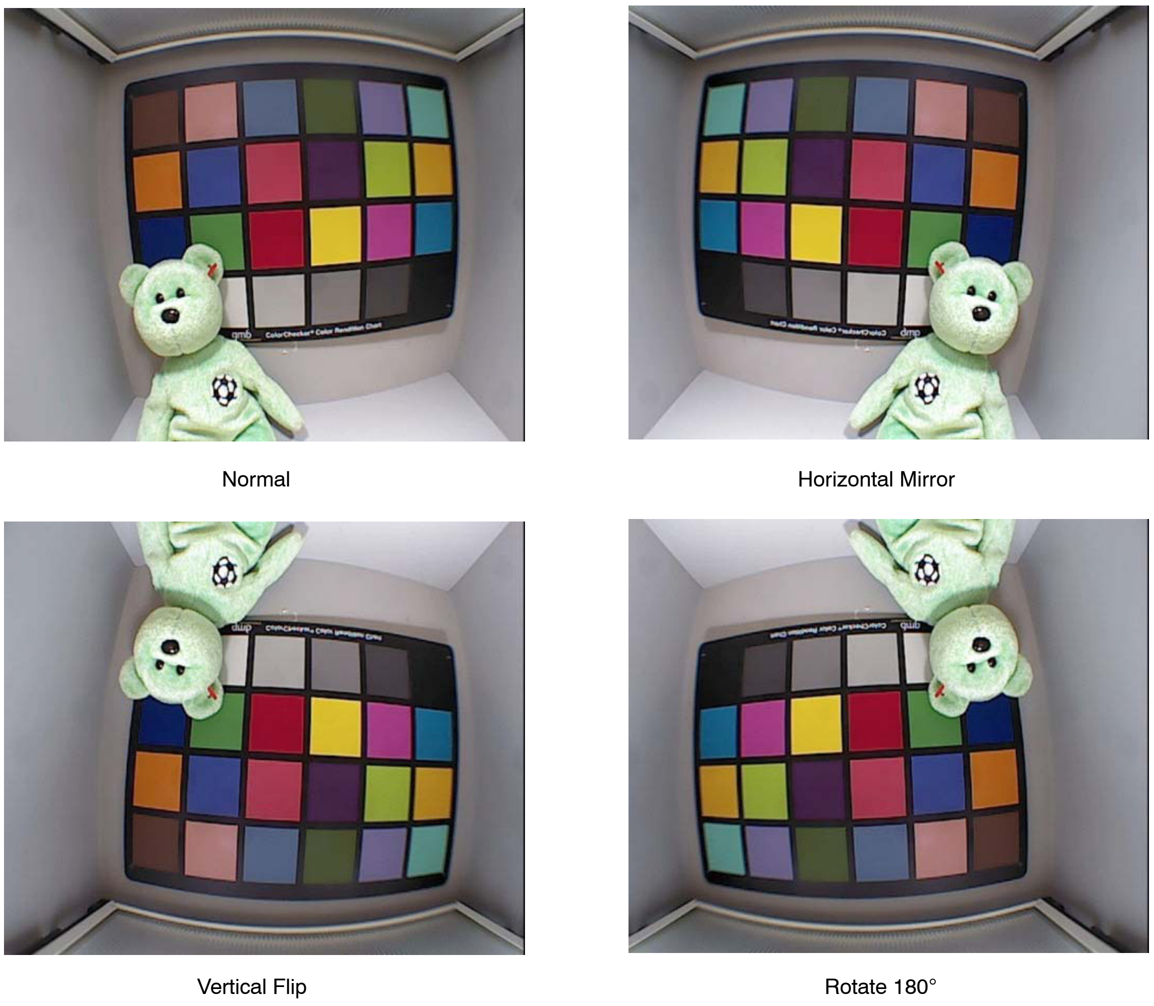


Figure 23. Different Image Orientations

Adjust Output Image Size in Progressive Mode

There are four registers that control the size of the sensor output. They are only used in progressive scan mode. Use the Register Wizard to generate settings for these four registers.

Table 17. REGISTERS CONTROLLING SIZE OF SENSOR OUTPUT

Map	Address	Name	Default Value	Descriptions
CamControl	0xC800	cam_sensor_cfg_y_addr_start	0x0024	The first row of visible pixels to be read out (not counting any dark columns that may be read). This value must be even.
CamControl	0xC802	cam_sensor_cfg_x_addr_start	0x0028	The first column of visible pixels to be read out (not counting any dark columns that may be read). This value must be even.
CamControl	0xC804	cam_sensor_cfg_y_addr_end	0x020B	The last row of visible pixels to be read out. This value must be odd.
CamControl	0xC806	cam_sensor_cfg_x_addr_end	0x02AF	The last column of visible pixels to be read out. This value must be odd.

Zoom and Pan

The ASX340AT supports zoom x1 and x2 modes, in interlaced and progressive scan modes. The progressive support is limited to the VGA60 or VGA50 modes.

In the zoom x2 modes, the sensor is configured for QVGA (320x240), and the zoom x2 window can be configured to pan around the VGA window.

The Pan feature can be used once the image is zoomed in as described above. To move (or Pan) the image view horizontally or vertically, a position offset is applied to the corresponding variables as shown in the table below. In DevWare, zoom and pan are accessible through Control Zoom/Pan Control.

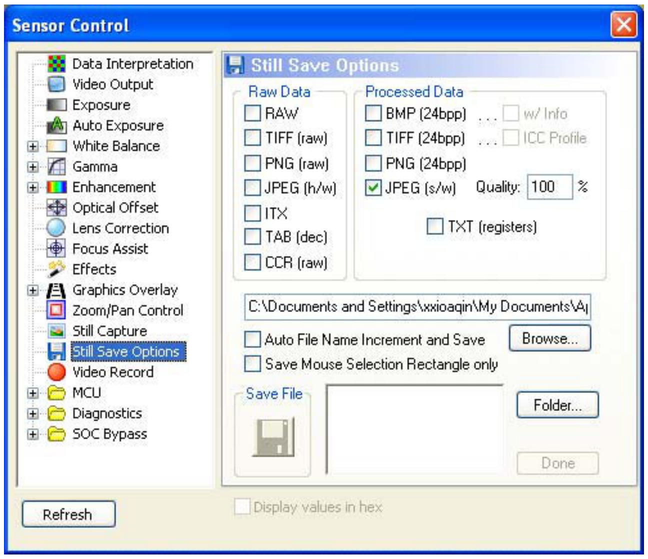
Table 18. REGISTERS CONTROLLING SIZE OF SENSOR OUTPUT

Map	Address	Name	Default Value	Descriptions
CAMControl	0xC864	cam_zoom_factor	0x01	0: Reserved. 1: x1 zoom. 2: x2 zoom. 3: Reserved.
CAMControl	0xC865	cam_zoom_y_start	0x78	Determines the starting row of the QVGA logical sensor ROI when in zoom x2 mode. A value of 0 indicates the topmost logical row.
CAMControl	0xC866	cam_zoom_x_start	0x00A0	Determines the starting column of the QVGA logical sensor ROI when in zoom x2 mode. A value of 0 indicates the leftmost logical column.

Still Image Capture

DevWare allows the user to take a still image capture using Capture button on the toolbar. Options such as image resolution, frames, or delays are configurable through *Control → Still Capture.*

Still image save options are configurable through *Control → Still Save Options* as shown in Figure 24. A variety of file formats can be chosen for saving the images, including formats for raw data as well as processed data. The TXT (registers) option allows the user to save a copy of all the register settings when a still image is captured. This is useful if register settings need to be compared for image appearance comparison and analysis. The still images can either be saved automatically with auto−incremented file names, or be saved manually using the *Save File* button after a specific name entered by the user in the directory bar.



Raw Data: [ ] RAW, [ ] TIFF (raw), [ ] PNG (raw), [ ] JPEG (h/w), [ ] ITX, [ ] TAB (dec), [ ] CCR (raw)

Processed Data: [ ] BMP (24bpp) ... [ ] w/ Info, [ ] TIFF (24bpp) ... [ ] ICC Profile, [ ] PNG (24bpp), [✓] JPEG (s/w) Quality: 100 %, [ ] TXT (registers)

C:\Documents and Settings\xxioaqin\My Documents\A

[ ] Auto File Name Increment and Save

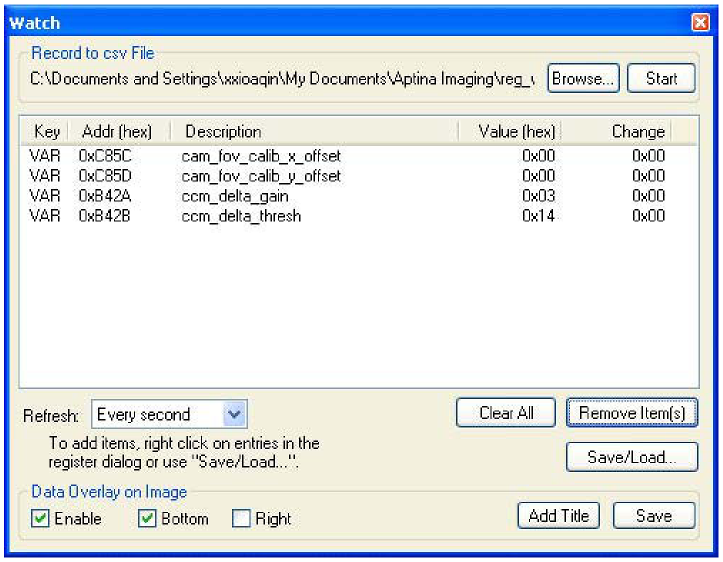
[ ] Save Mouse Selection Rectangle only

[ ] Display values in hex

Figure 24. Still Save Options

Register and/or variable values can be overlaid on the image to be saved as well. The following procedure outlines how to do this:

- First, add the registers/variables to Watch
- On Watch GUI as shown in Figure 25, check *Enable* and the location to show the registers/variables on the image in Data Overlay on Image section.
- A title can also be added to show on the image.



Watch

Record to csv File

C:\Documents and Settings\xxioaqin\My Documents\Aptina Imaging\reg_\

Key	Addr (hex)	Description	Value (hex)	Change
VAR	0xC85C	cam_fov_calib_x_offset	0x00	0x00
VAR	0xC85D	cam_fov_calib_y_offset	0x00	0x00
VAR	0xB42A	ccm_delta_gain	0x03	0x00
VAR	0xB42B	ccm_delta_thresh	0x14	0x00

Refresh: Every second

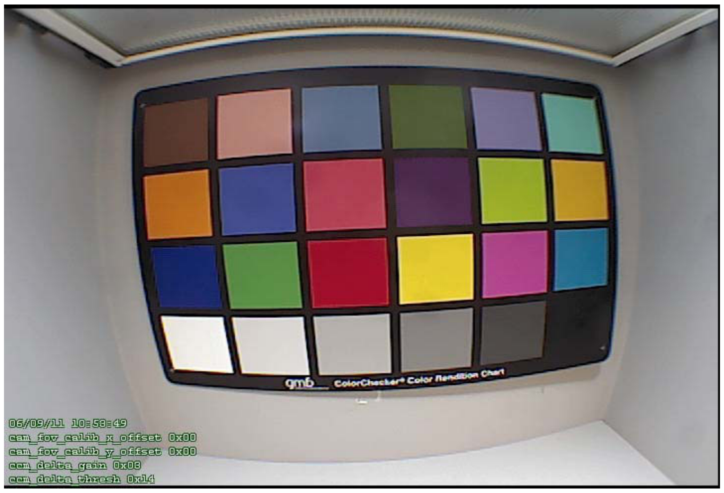
To add items, right click on entries in the register dialog or use "Save/Load...".

Data Overlay on Image

[✓ Enable] [✓ Bottom] ☐ Right

Figure 25. Watch GUI to Enable Register or Variable Overlay on Image

An example is shown below.



```
06/09/11 10:58:49
cam_fov_calib_x_offset 0x00
cam_fov_calib_y_offset 0x00
ccm_delta_gain 0x03
ccm_delta_thresh 0x14
```

Figure 26. Still Image Captured with Register or Variable Overlay

Video Capture

DevWare allows the user to record video through *Control →Video Record.* The video can be recorded either in RAW or AVI formats.

LENS SHADING CORRECTION

This section outlines the procedure for performing lens shading correction (LSC) calibration. Camera module/lens will usually have some signal degradation on the sensor periphery due to optical and geometrical factors. Lens shading correction compensates for the signal degradation by digitally gaining pixels on the periphery. In ASX340AT, the lens shading correction function is performing on all four color channels − Red, GreenR, Blue, and GreenB. The result of the procedure is sets of coefficients being generated that can be loaded to the sensor by either host mode or flash mode operation.

The lens shading correction calibration is built into **onsemi** Development Software (DevWare). To perform the lens shading correction, a uniformly illuminated light source is required to generate a flat field for the calibration. The flat field light intensity variation must be no more than two percent over the entire FOV at a color temperature of 5000 K, for example (the user can choose a different color temperature that best suits each individual application). The light source and sensor system must be shielded from other light sources. Figure 27 depicts the setting. The sensor (left), the diffuser (middle) and the light source (right) are arranged in a way where their center point are aligned on a straight line, and their surfaces are parallel to each other. The distance between the light source and the diffuser, and between the diffuser and the sensor, must be as short as possible to avoid reflections.

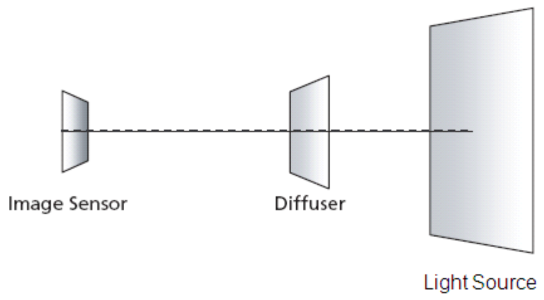


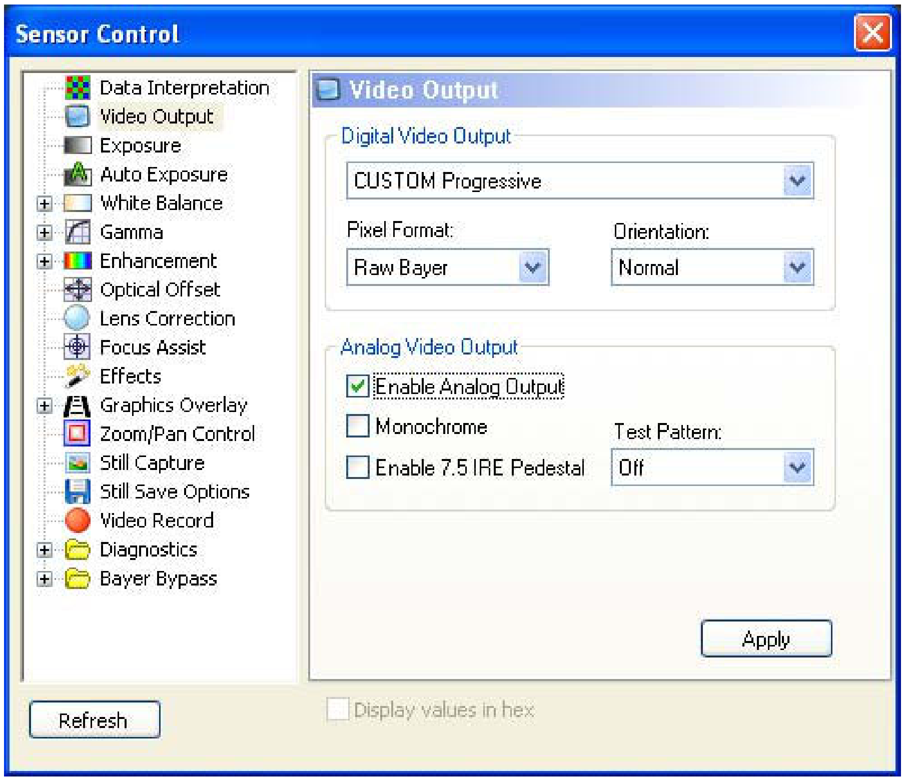
Figure 27. Lens Shading Correction Setup

Calibration Procedure

After the setup in Figure 27 is completed, the calibration procedure is outlined below:

1. Start DevWare and point the camera to a nearby object, about the same distance as the diffuser.
2. Adjust the focus.
3. Set cam_mode_select to 1, and do a change−config afterwards. Or directly load the preset *Enter Lens Calibration* in the sensor's ini file.
4. The video output mode is now set to raw Bayer as shown in Figure 28.
5. Enable the *Row Marker and Graph*, and move the row marker to the center of the image. See Figure 29. The intensity graph shows pre−calibration status.
6. Increase the integration time, i.e. coarse_integration_time (R0x3012 on Core page) to get the RGB output level between 200 to 220 as shown in Figure 30.
7. Go to *Control* → *Lens Correction* as shown in Figure 31:
    1. If *Enable* is checked, uncheck it to disable lens correction.
    2. Click on *Find Optical Center.*
    3. Set the calibration falloff to 90~100% (for wide angle lens, this may need to be reduced in order to achieve the uniformed brightness with moderate amount of noise in the corners)
    4. The lens radius parameter is 0 by default, meaning the whole image will be used for calibration. For wide angle lens, lens radius could be configured to exclude dark areas on the edges. Lens radius is specified in pixels. For example, if lens radius is set to 200, it means only pixels inside the internal cut square of the circle with radius 200 (centered in the image) will be used for lens correction. The internal cut square is allowed to be outside of the image, in which case, only the real pixels will be used for lens correction. Click on *Calibrate Lens Correction.*
    5. Click on ***Save as** Ini*.

The saved INI file contains the settings for lens shading correction.



Digital Video Output: CUSTOM Progressive

Pixel Format: Raw Bayer

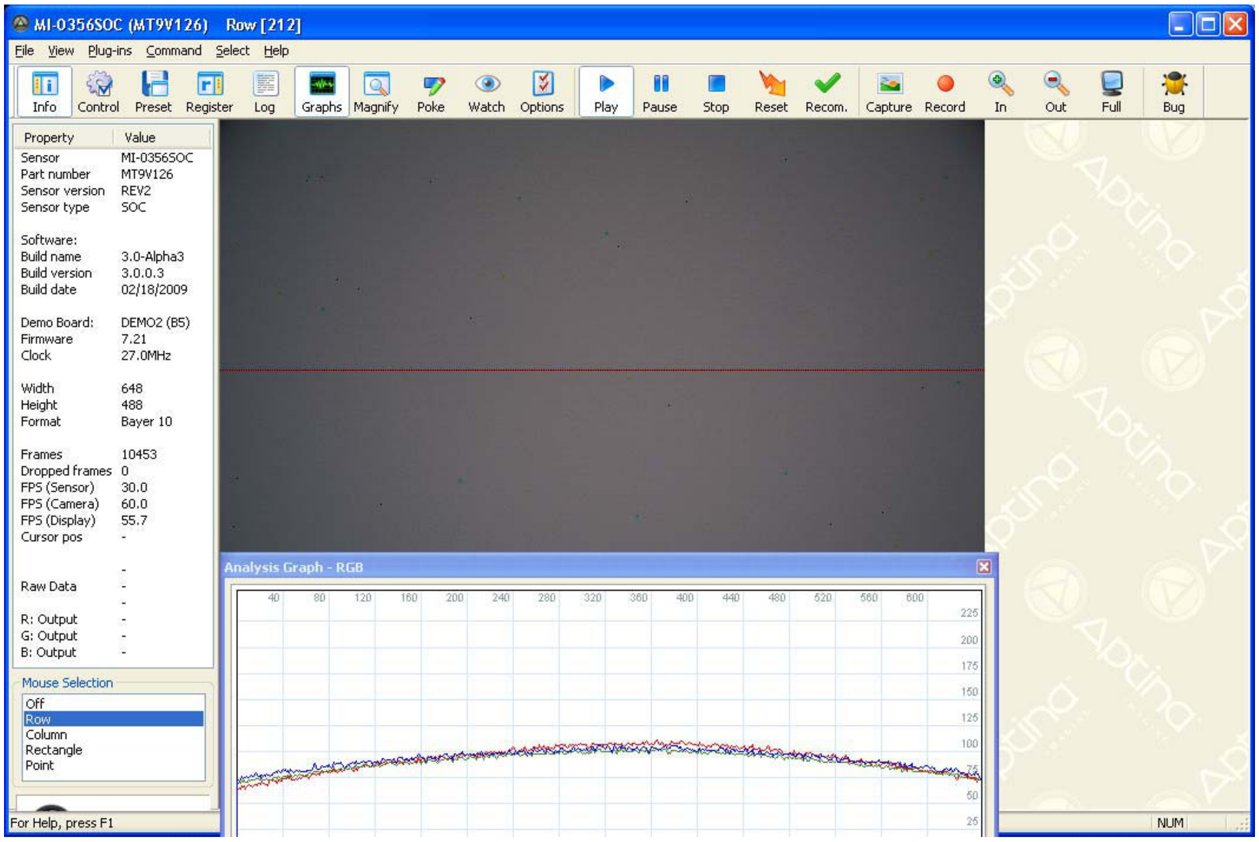
Orientation: Normal

Analog Video Output

- [✓] Enable Analog Output
- [ ] Monochrome
- [ ] Enable 7.5 IRE Pedestal

Test Pattern: Off

Figure 28. Video Output Configuration for Lens Shading Correction Calibration


MI-0356SOC (MT9V126) Row [212]
File View Plug-ins Command Select Help
Info Control Preset Register Log Graphs Magnify Poke Watch Options Play Pause Stop Reset Recom. Capture Record In Out Full Bug
Analysis Graph - RGB
40 80 120 160 200 240 280 320 360 400 440 480 520 560 600
225 200 175 150 125 100 75 50 25
Mouse Selection
Off
Row
Column
Rectangle
Point
For Help, press F1
NUM

Property	Value
Sensor	MI-0356SOC
Part number	MT9V126
Sensor version	REV2
Sensor type	SOC
Software:	
Build name	3.0-Alpha3
Build version	3.0.0.3
Build date	02/18/2009
Demo Board:	DEMO2 (B5)
Firmware	7.21
Clock	27.0MHz
Width	648
Height	488
Format	Bayer 10
Frames	10453
Dropped frames	0
FPS (Sensor)	30.0
FPS (Camera)	60.0
FPS (Display)	55.7
Cursor pos	-
	-
Raw Data	-
	-
R: Output	-
G: Output	-
B: Output	-

Figure 29. Intensity Graph after Increasing Integration Time

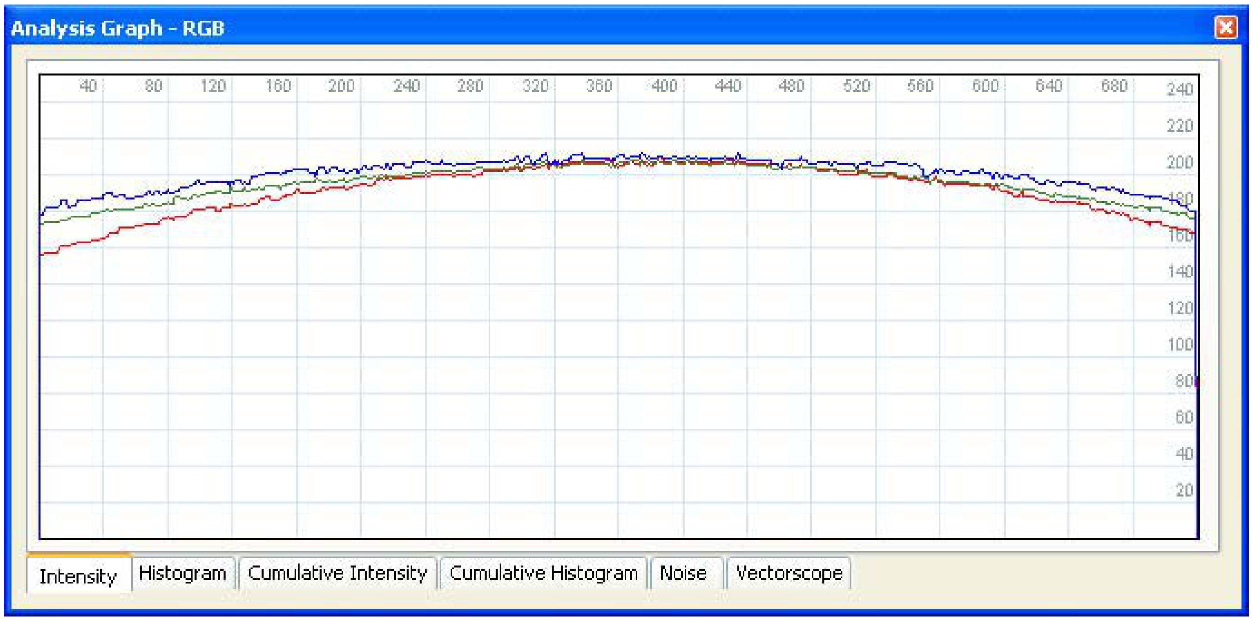
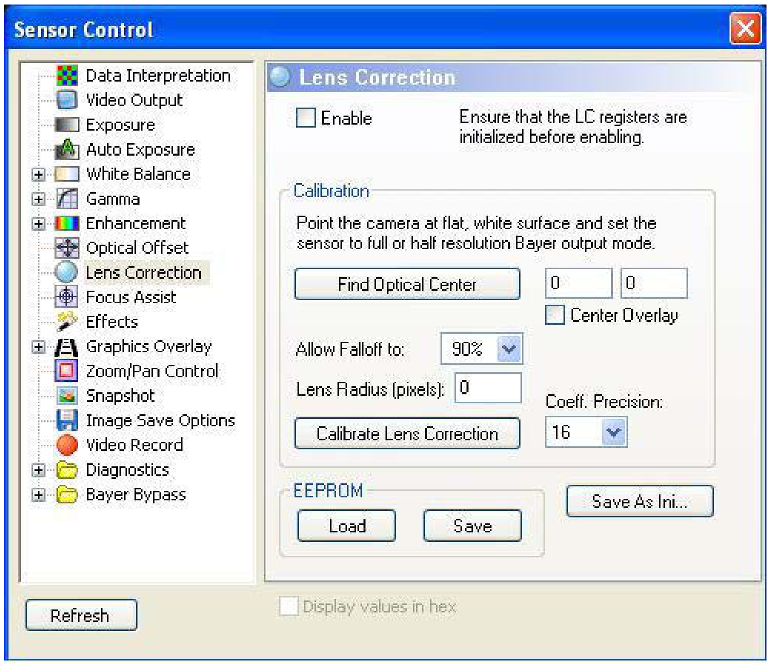


Figure 30. Intensity Graph after Increasing Integration Time



Lens Correction

☐ Enable

Ensure that the LC registers are initialized before enabling.

Calibration

Point the camera at flat, white surface and set the sensor to full or half resolution Bayer output mode.

Find Optical Center: 0 0

☐ Center Overlay

Allow Falloff to: 90%

Lens Radius (pixels): 0

Coeff. Precision: 16

EEPROM

☐ Display values in hex

Figure 31. Lens Correction GUI

A saved raw image can also be used for lens shading correction. The raw image needs to be in a raw Bayer format. The same LSC calibration procedure applies.

AUTO EXPOSURE

The auto exposure (AE) algorithm performs automatic adjustments of the image brightness by controlling the exposure time and analog gains of the sensor core, as well as the digital gains applied to the image.

AE is implemented by means of a firmware driver that analyzes image statistics (for example, average luma) collected by the exposure measurement engine, decides the best exposure and gain settings, and programs the sensor core and color pipeline accordingly.

When using the ASX340AT, AE starts by reading the average luma (e.g. ae_rule_avg_y_from_stats) and compa ring it with target luma (e.g. cam_aet_target_average_luma) to decide if exposure needs to be increased or decreased. Details on different AE configurations are provided in the following sections.

AE Algorithms

AE algorithms determine the methodology of calculating the average brightness/luma of the current scene. Table 19 includes the key registers/variables for AE algorithm controls. Four auto exposure algorithms are available. AE_RULE_ALGO (Rx0A404) controls the selection of auto exposure algorithm used:

1. Average brightness tracking (ABT) (ae_rule_algo VAR = 9, 0x0004, 0x0000 or REG = 0xA404, 0x0000)
   The average brightness tracking AE uses a constant average tracking algorithm where a target brightness value is compared to a current brightness value, and the gain and integration time are adjusted accordingly to meet the target requirement.
2. Weighted Average Brightness (ae_rule_algo VAR = 9, 0x0004, 0x0001 or REG = 0xA404, 0x0001)
   Each of the 25 windows can be assigned a weight relative to other window weights, which can be changed independently of each other. For example, the weights can be set to allow the center of the image to be weighted higher than the periphery. See Figure 32.

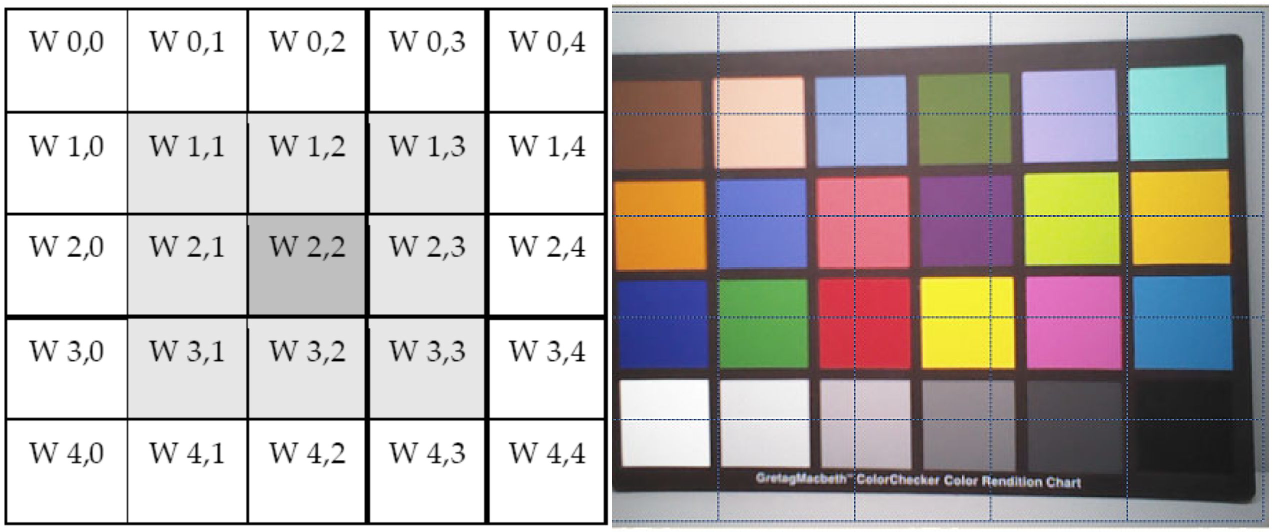


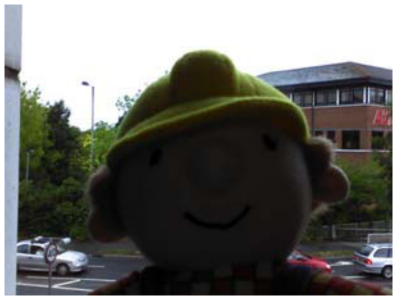
Figure 32. 5 x 5 Grid

3. Adaptive Weighted AE for highlights (ae_rule_algo VAR = 9, 0x0004, 0x0002 or REG = 0xA404, 0x0002)
   The scene will be exposed based on the brightness of each window, and will adapt to correctly expose the highlights (brighter windows). This will correctly expose the foreground of an image when the background is dark.
4. Adaptive Weighted AE for lowlights (ae_rule_algo VAR = 9, 0x0004, 0x0003 or REG = 0xA404, 0x0003)
   The scene will be exposed based on the brightness of each window, and will adapt to correctly expose the lowlights. This will correctly expose the foreground of an image when the background is brighter.

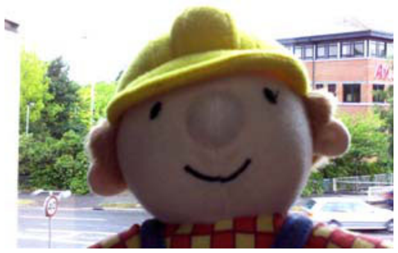
Sample images below show the performance of the different AE algorithms.

Light Background

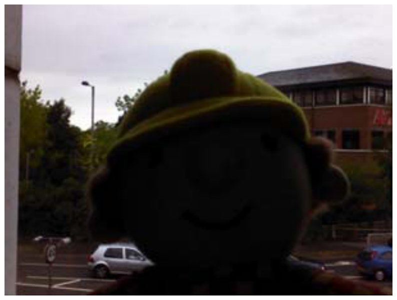
Average Brightness Tracking or Average Y



Weighted Average Brightness (centre)



Adaptive weighted based on zone luma (highlights)



Adaptive weighted based on zone luma (lowlights)

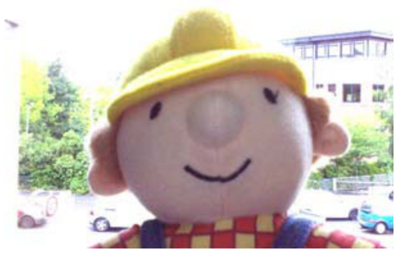
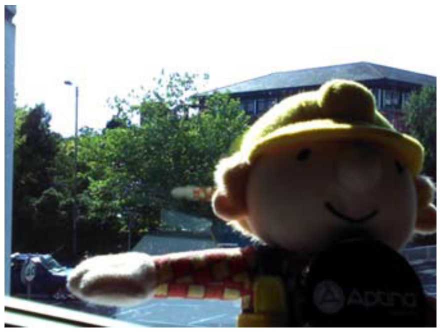


Figure 33.

In the use case above, the Adaptive weighted for lowlights exposes the face slightly better when compared to the Weighted Average Brightness.

However, if the foreground subject is moved off−center:

Weighted Average Brightness (centre) (lowlights)



Adaptive weighted based on zone luma

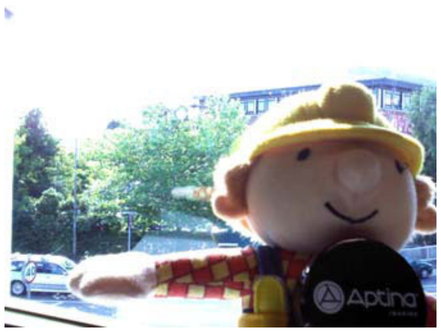
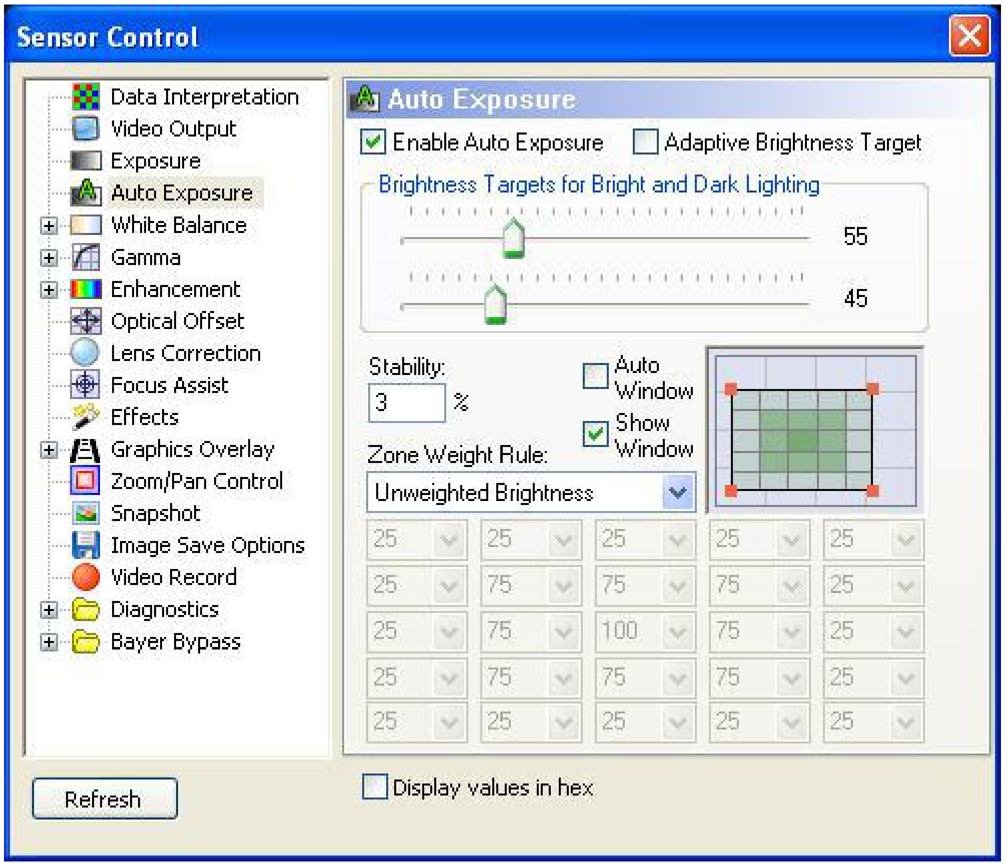


Figure 34.

This shows the advantage of using the Adaptive Weighted AE for lowlights (ae_rule_algo = 0x03); when the face moves off center it still is exposed correctly.

In DevWare, on *Control* → *Auto Exposure* as shown in Figure 33, the AE algorithms are configurable under Zone Weight Rule. The weighted tables are also configurable on the same GUI.



[✓] Enable Auto Exposure
[ ] Adaptive Brightness Target

Brightness Targets for Bright and Dark Lighting: 55, 45

Stability: 3 %

[ ] Auto Window
[✓] Show Window

Zone Weight Rule: Unweighted Brightness

25	25	25	25	25
25	75	75	75	25
25	75	100	75	25
25	75	75	75	25
25	25	25	25	25

[ ] Display values in hex

Figure 35. AE Algorithm Configuration in DevWare

Table 19. VARIABLES ASSOCIATED WITH AE ALGORITHM SETUP

Address	Name	Bits	Function
0xA404	ae_rule_algo	[1:0]	Selects AE algorithm used.
			0: Average Brightness.
			1: Weight Brightness.
			2: Adaptive for Highlight.
			3: Adaptive for Lowlights.
0xA407	ae_rule_ae_weight_table_0_0	[7:0]	AE weight 0.0
0xA408	ae_rule_ae_weight_table_0_1	[7:0]	AE weight 0.1
0xA409	ae_rule_ae_weight_table_0_2	[7:0]	AE weight 0.2
0xA40A	ae_rule_ae_weight_table_0_3	[7:0]	AE weight 0.3
0xA40B	ae_rule_ae_weight_table_0_4	[7:0]	AE weight 0.4
0xA40C	ae_rule_ae_weight_table_1_0	[7:0]	AE weight 1.0
0xA40D	ae_rule_ae_weight_table_1_1	[7:0]	AE weight 1.1
0xA40E	ae_rule_ae_weight_table_1_2	[7:0]	AE weight 1.2
0xA40F	ae_rule_ae_weight_table_1_3	[7:0]	AE weight 1.3
0xA410	ae_rule_ae_weight_table_1_4	[7:0]	AE weight 1.4
0xA411	ae_rule_ae_weight_table_2_0	[7:0]	AE weight 2.0
0xA412	ae_rule_ae_weight_table_2_1	[7:0]	AE weight 2.1
0xA413	ae_rule_ae_weight_table_2_2	[7:0]	AE weight 2.2
0xA414	ae_rule_ae_weight_table_2_3	[7:0]	AE weight 2.3

Table 19. VARIABLES ASSOCIATED WITH AE ALGORITHM SETUP (continued)

Address	Name	Bits	Function
0xA415	ae_rule_ae_weight_table_2_4	[7:0]	AE weight 2.4
0xA416	ae_rule_ae_weight_table_3_0	[7:0]	AE weight 3.0
0xA417	ae_rule_ae_weight_table_3_1	[7:0]	AE weight 3.1
0xA418	ae_rule_ae_weight_table_3_2	[7:0]	AE weight 3.2
0xA419	ae_rule_ae_weight_table_3_3	[7:0]	AE weight 3.3
0xA41A	ae_rule_ae_weight_table_3_4	[7:0]	AE weight 3.4
0xA41B	ae_rule_ae_weight_table_4_0	[7:0]	AE weight 4.0
0xA41C	ae_rule_ae_weight_table_4_1	[7:0]	AE weight 4.1
0xA41D	ae_rule_ae_weight_table_4_2	[7:0]	AE weight 4.2
0xA41E	ae_rule_ae_weight_table_4_3	[7:0]	AE weight 4.3
0xA41F	ae_rule_ae_weight_table_4_4	[7:0]	AE weight 4.4

AE Window

The AE stats window can be moved by the user to determine where the stats (for example, the average luma) are gathered in the image for auto exposure calculations.

AE window can be set by directly writing to the variables in Table 20. These variables define the first window of the 5x5 AE grids as shown in Figure 36.

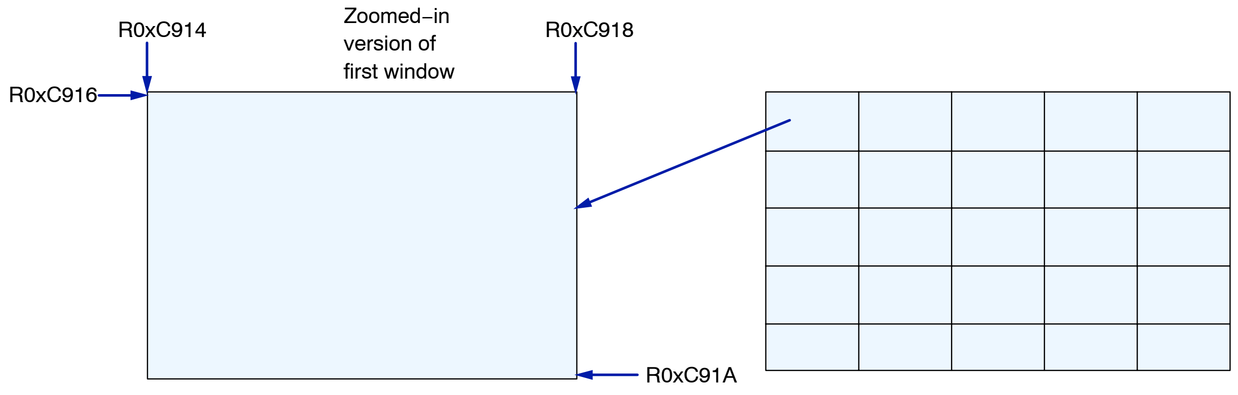
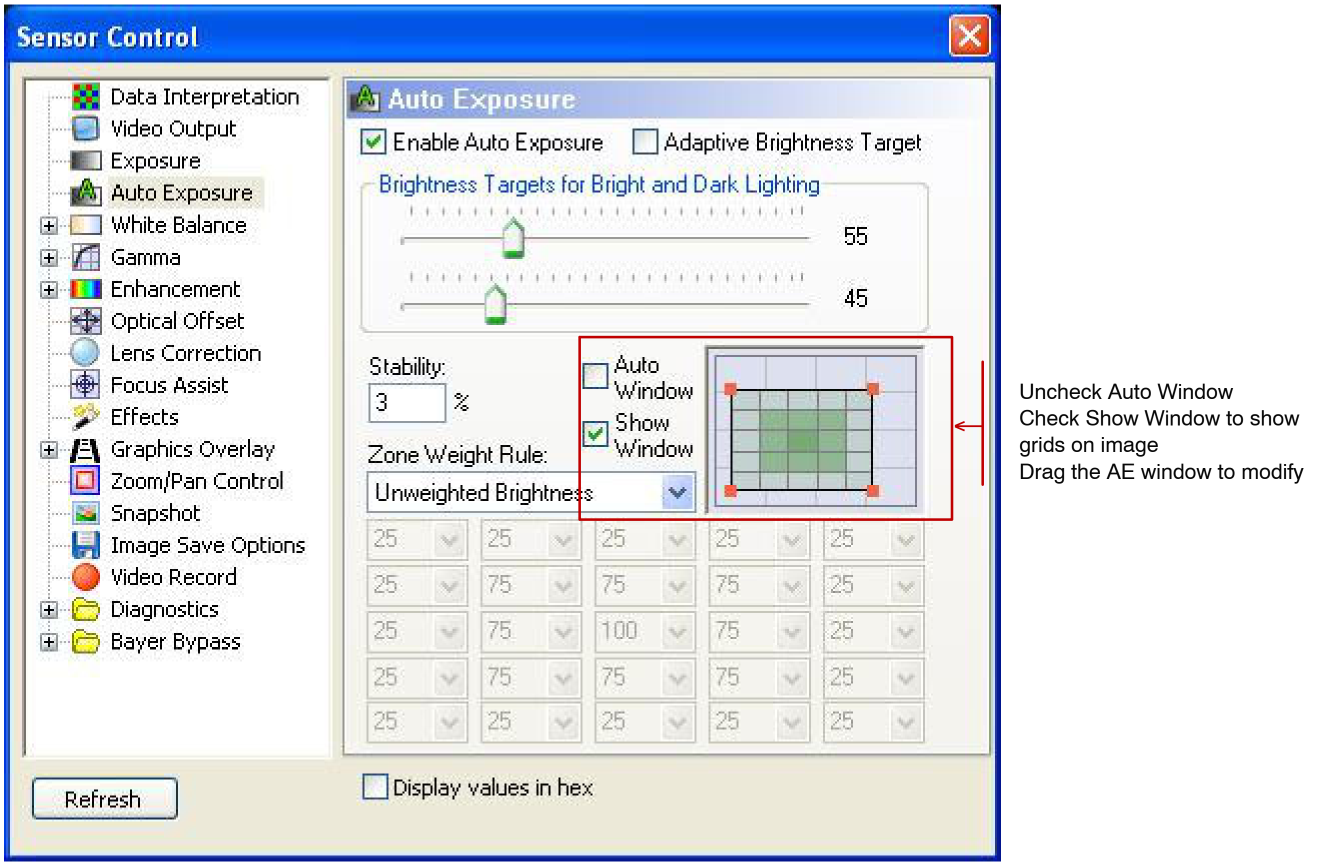


Figure 36. AE Window

Users are also able to specify AE window through *Control →Auto Exposure* as shown in Figure 37.



[✓] Enable Auto Exposure [ ] Adaptive Brightness Target

Brightness Targets for Bright and Dark Lighting: 55, 45

Stability: 3 %

Zone Weight Rule: Unweighted Brightness

[ ] Auto Window [✓] Show Window

25	25	25	25	25
25	75	75	75	25
25	75	100	75	25
25	75	75	75	25
25	25	25	25	25

Uncheck Auto Window
Check Show Window to show grids on image
Drag the AE window to modify

Figure 37. Change AE Window on Control − Auto Exposure

Table 20. VARIABLES USED FOR AE HISTOGRAM WINDOW

Address	Name	Bits	Function
0XC914	cam_stat_ae_intial_window_xstart	[15:0]	Start pixel for the X−coordinate of AE histograms window. This is only the first window of the 5x5 grid.
0XC916	cam_stat_ae_initial_window_ystart	[15:0]	Start pixel for the Y−coordinate of AE histograms window. This is only the first window of the 5x5 grid.
0Xc918	am_stat_ae_initial_window_xend	[15:0]	End pixel for the X−coordinate of AE histograms window. This is only the first window of the 5x5 grid.
0xC91A	cam_stat_ae_initial_window_yend	[15:0]	End pixel for the Y−coordinate of AE histograms window. This is only the first window of the 5x5 grid.

AE Modes

Three different AE modes are available for ASX340AT: Indoor AE, Discrete Frame Rate, and Adaptive AE Target. These three modes are independent, and can be enabled at the same time. When all these modes are disabled, the AE will simply function based on the lighting conditions discussed in“AE Track”. Table 21 includes the key variables for AE mode configuration.

Indoor AE

When R0xC86C[0] = 0x1 the AE will have a restriction placed on it to ensure that the integration time will always remain a multiple of the flicker frequency. This ensures that flicker will be avoided at all times. In this mode the integration time will be limited to at least 1 flicker period. Due to this limitation, under bright light conditions, images can be overexposed.

Discrete Frame Rate

This feature is only applicable to progressive scan modes as interlaced modes (i.e. NTSC or PAL) require a fixed frame rate. To enable this feature, the user should set R0xC86C[1]=0x1 and then execute a change−config command (Note: Likewise, when users wish to turn this mode off, they set the bit = 0 and execute a Change−Config command).

When this mode is being used, as the scene illumination decreases, the frame rate will always decrease by half, so it is important that the user ensures that the frame rates are programmed correctly or it will round down to the nearest one. Similarly, as the scene illumination increases the frame rate will increase by factors of two. Refer to “Variable Frame Rates” for more details.

Adaptive AE Target

The Adaptive AE feature for the ASX340AT allows the Average Brightness (Luma) to be set to different levels depending on light levels. In other words, this allows setting a different AE target under low light conditions versus bright light conditions.

The method of setting the values in bright light and low light is identical. They depend on the value of the cam_ll_inv_brightness_metric to select where the transition from the first parameter to the second starts and ends (the values in between are linearly interpolated).

The relationship between the firmware variables is shown below.

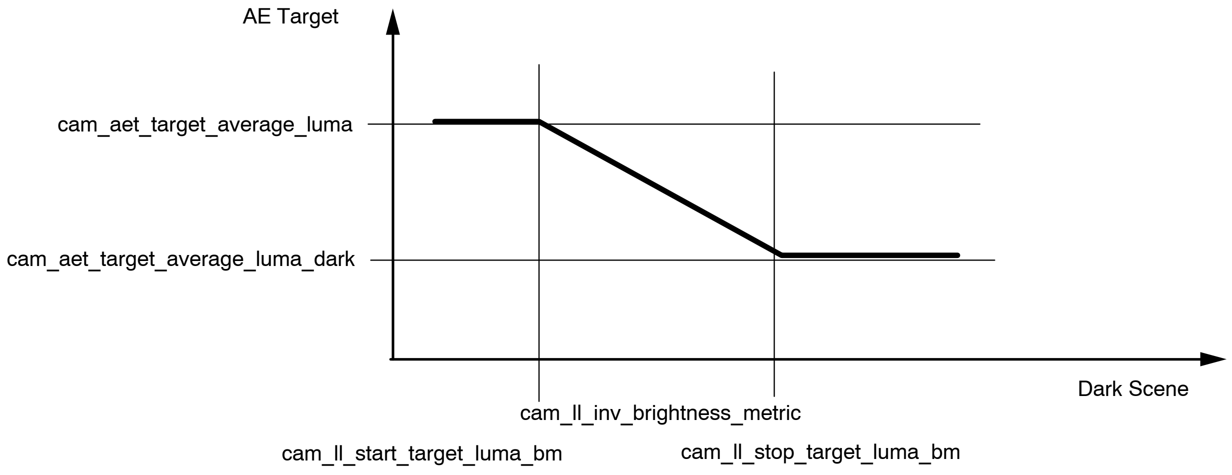

AE Target
cam_aet_target_average_luma
cam_aet_target_average_luma_dark
Dark Scene
cam_ll_inv_brightness_metric
cam_ll_start_target_luma_bm
cam_ll_stop_target_luma_bm

Figure 38. AE Adaptive Dampening Diagram

1. The cam_aet_target_average_luma value can be greater or less than cam_aet_target_average_luma_dark value.
2. The cam_ll_start_target_luma_bm value must be less than cam_ll_stop_target_luma_bm value.

To set up the cam_ll_start_target_luma_bm and cam_ll_stop_target_luma_bm points:

1. Set camera up for desired frame rate and with 100lux light.
2. Note the value of cam_ll_inv_brightness_metric; put this into cam_ll_start_target_luma_bm.
3. Change light level to 20 lux.
4. Note the value of cam_ll_inv_brightness_metric; put this into cam_ll_stop_target_luma_bm.

The choice of light levels of 100 and 20 lux are a guide and the actual values used are at the discretion of the user.

Table 21. VARIABLES FOR AE MODE SETUP

Address	Name	Bits	Function
0xC86C	cam_aet_aemode	[2:0]	Selects different AE modes: Bit [0]: 1 Indoor AE is selected Bit [1]: 1 Discrete Frame Rate is selected Bit [2]: 1 Adaptive AE Target
0xC86E	cam_aet_target_average_luma	[7:0]	target average brightness when cam_ll_inv_brightness_metric < cam_ll_start_target_luma_bm (when Adaptive AE Target mode is enabled)
0xC86F	cam_aet_target_average_luma_dark	[7:0]	target average brightness when cam_ll_inv_brightness_metric > cam_ll_start_target_luma_bm_dark (when Adaptive AE Target mode is enabled)

Table 21. VARIABLES FOR AE MODE SETUP(continued)

Address	Name	Bits	Function
0xC954	cam_ll_start_target_luma_bm	[15:0]	Start value of cam_ll_inv_brightness_metric for adaptive target luma AE mode
0xC956	cam_ll_stop_target_luma_bm	[15:0]	Stop value of cam_ll_inv_brightness_metric for adaptive target luma AE mode
0xA807	ae_track_target_average_luma	[7:0]	The average brightness target that AE_Track is trying to maintain. This is being controlled by target average luma on Cam page.
0xA808	ae_track_gate_percentage	[7:0]	Hysteresis gate around target brightness, expressed as a percentage of target brightness.
0xA809	ae_track_current_average_luma	[7:0]	Current average brightness that is being measured in the scene.

Variable Frame Rates

ASX340AT supports variable frame rates only in custom−progressive mode with the following settings:

- cam_frame_scan_mode = 1 (i.e., R0xC858[0] = 1)
- cam_frame_scan_progressive_mode = 2 (i.e. R0XC858[4:3] = 2)

Variable frame−rate mode is then enabled by configuring CAM_AET_MIN_FRAME_RATE to be less than CAM_AET_MAX_FRAME_RATE (which is controlled by CAM_SENSOR_CFG_FRAME_LENGTH_LINES and CAM_SENSOR_CFG_LINE_LENGTH_PCK).

ASX340AT supports two variable frame rate modes.

1. Continuous Frame Rate:
   This mode is selected by CAM_AET_DISCRETE_FRAMERATE = 0 (R0xC86C[1] = 0). In this mode, the FW varies the frame rates in multiples of the flicker period dependent upon the scene luma. The maximum frame rate is as configured by RegWizard (reported by the variable CAM_AET_MAX_FRAME_RATE), and the minimum frame rate is controlled by CAM_AET_MIN_FRAME_RATE.
   For example, if maximum frame rate is 30 fps, and minimum frame rate is 15 fps, and the flicker frequency is 60 Hz, the firmware will select the frame rates shown in Table 22:

Table 22. EXAMPLE − CONTINUOUS FRAME RATE

Number of Flicker Periods (*8.33 ms)	Frame Period (ms)	Frame Rate
1−4	33.33	30
5	41.66	24
6	50	20
7	58.33	17.14
8	66.66	15

11. If the minimum frame rate is not a multiple of the flicker period, the firmware will use the nearest frame rate that is a multiple of the flicker period, but not less than the configured minimum frame rate. In the example above, if the minimum frame rate was 16, the firmware will stop at 17.14 fps.

2. Discrete Frame Rate:
   This mode is selected by CAM_AET_DISCRETE_FRAMERATE = 1 (R0xC86C[1] = 1). In this mode, the FW varies the frame rates in factors of two, i.e. doubling or halving the frame rate also in multiples of the flicker period dependent upon the scene luma. The maximum frame rate is as configured by RegWizard (reported by the variable CAM_AET_MAX_FRAME_RATE), and the minimum frame rate is controlled by CAM_AET_MIN_FRAME_RATE. or example, if maximum frame rate is 60 fps, and minimum frame rate is 7.5 fps, and the flicker frequency is 60 Hz, the firmware will select the following frame rates:

Table 23. EXAMPLE – DISCRETE FRAME RATE

Number of Flicker Periods (*8.33 ms)	Frame Period (ms)	Frame Rate
1–2	16.66	60
3–4	33.33	30
5–8	66.66	15
9–16	133.33	7.5

12. If the maximum frame rate is not a multiple of the flicker period, the firmware will choose the nearest frame rate that is a multiple of the flicker period. This will result in a slower max frame rate than selected.

AE Adaptive Dampening

The AE adaptive dampening feature adjusts the amount of dampening applied to AE while moving towards target luma value. When the target luma is far away, the adaptive dampening algorithm tries to settle AE in fewer frames.

ae_track_ae_dampening_speed is a read–only variable describing the current amount of dampening applied to AE while moving towards target luma value (that is, *ae_track_target_average_luma*). This variable ranges between 0 and 32. A higher dampening speed helps the AE take fewer frames to stabilize. For example, 32 means there is no dampening, and therefore AE will take the full speed and try to settle in a single frame; 16 means that the AE will take half the speed and try to move to the mid–point between current and target luma in each step.

This variable is calculated by firmware dependent on the setup of ae_track_start_adapt_dampening_speed and ae_track_stop_adapt_dampening_speed:

- When ae_track_start_adapt_dampening_speed = ae_track_stop_adapt_dampening_speed, ae_track_ae_dampening_speed is fixed, and it is equal to ae_track_start_adapt_dampening_speed;
- When ae_track_start_adapt_dampening_speed < ae_track_stop_adapt_dampening_speed, AE adaptive dampening feature is turned on. In this case, ae_track_ae_dampening_speed is updated according to Figure 39.
    - When ae_track_estimated_gain < ae_track_start_adapt_dampening_gain, ae_track_ae_dampening_speed = ae_track_start_adapt_dampening_speed;
    - When ae_track_estimated_gain is in between ae_track_start_adapt_dampening_gain and ae_track_stop_adapt_dampening_gain, ae_track_ae_dampening_speed is calculated based on the linear interpolation between ae_track_start_adapt_dampening_speed and ae_track_stop_adapt_dampening_speed;
    - When ae_track_estimated_gain > ae_track_stop_adapt_dampening_gain, ae_track_ae_dampening_speed = ae_track_stop_adapt_dampening_speed;

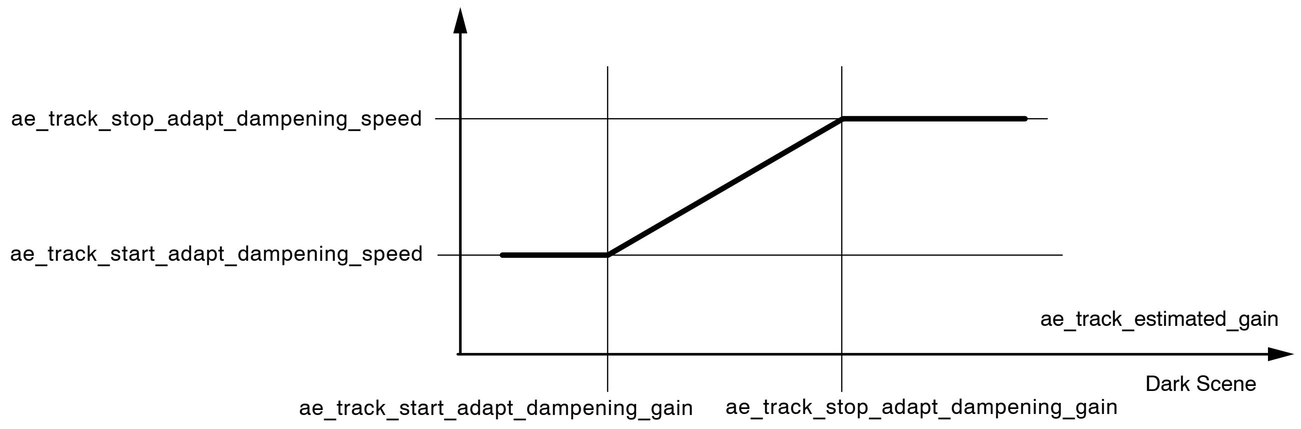


Figure 39. AE Adaptive Dampening Diagram

This feature is recommended to take the default values as shown in Table 24. However, the user may consider to adjust ae_track_start_adapt_dampening_speed for customization.

Table 24. VARIABLES FOR AE ADAPTIVE DAMPENING

Address	Name	Bits	Function
0xA852	ae_track_start_adapt_dampening_gain	[15:0]	Controls the start threshold for adaptive dampening.
0xA854	ae_track_stop_adapt_dampening_gain	[15:0]	Controls the stop threshold for adaptive dampening.
0xA856	ae_track_start_adapt_dampening_speed	[7:0]	Controls the start dampening speed of adaptive dampening. 0: maximum dampening 32: no dampening
0xA857	ae_track_stop_adapt_dampening_speed	[7:0]	Controls the stop dampening speed of adaptive dampening.
0xA80B	e_track_ae_dampening_speed	[7:0]	Read−only. Indicates the current dampening being applied to the calculated change in virtual exposure.
0xA84E	ae_track_estimated_gain	[15:0]	Read−only. Indicates the (dampened) estimated change in virtual exposure that is required to settle AE.
0xA807	ae_track_target_average_luma	[7:0]	Read−only. The current average brightness target that AE_track is trying to maintain. It is controlled by target average luma on Cam page.
0xA809	ae_track_current_average_luma	[7:0]	Read−only. Current average brightness.

AE Track

Depending on the lighting conditions, the AE may settle in any of the two zones (indicated by ae_track_zone).

- Zone 0

  This zone is for bright conditions. When in this zone, the integration time is less than 1 flicker period and analog and digital gains are unity. Flicker may occur in this zone. If R0xC86C[0] = 0x1, i.e. indoor mode, AE will not go into zone 0.
- Zone 1

  This zone is for indoor lighting conditions. The integration time is kept as a multiple of the flicker period by controlling ae_track_fdzone. Analog gain and digital gains are permitted to reach their maximum values.

Figure 40 illustrates the various aspects of zone 0 and zone 1.

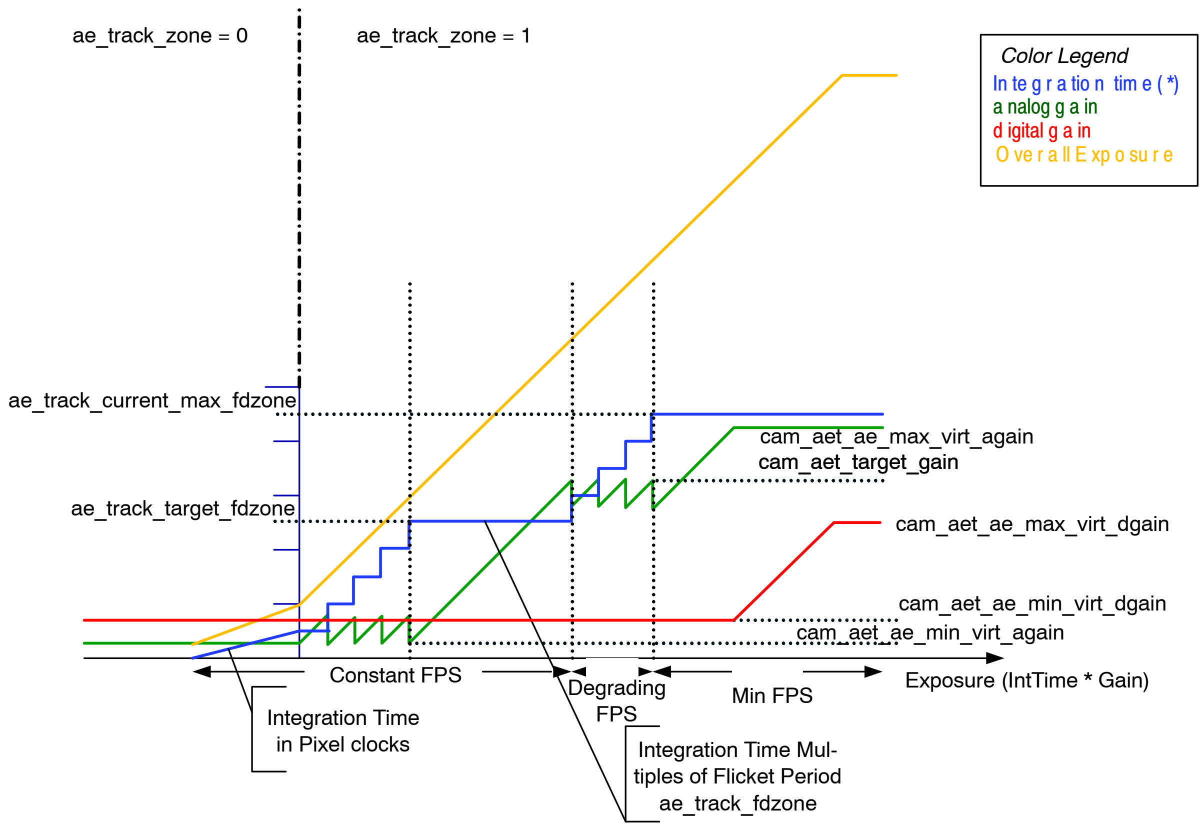


Figure 40. AE Track Zones

Table 25. VARIABLES FOR AE MODE SETUP

Address	Name	Bits	Function
0xA810	ae_track_current_max_fdzone	[15:0]	Read−only. The maximum number of flicker periods (of integration time) that AE track is permitted to use. This determines the minimum frame rate.
0xA814	ae_track_target_fdzone	[15:0]	Read−only. AE track tries to keep ae_track_fdzone below this value. When ae_track_fdzone reaches this value, AE track will then increase analog gain until cam_aet_target_gain is reached. AE track then increases ae_track_fdzone and analog gains until they both reach their maximum values.
0xA818	ae_track_fdzone	[15:0]	Read−only.The current number of flicker periods (of integration time) that AE track is using.
0xA81B	ae_track_zone	[7:0]	Read−only. The current AE track zone Bit[0]: 0 Zone 0 Bit[0]: 1 Zone 1
0xA83C	ae_track_virt_gain	[31:0]	Read−only. The current virtual gain (total of analog and digital gains), in units of 1/128.
0xC886	cam_aet_target_gain	[15:0]	The target analog gain. This value is used by AE Track to determine the maximum gain before starting to reduce the frame rate. The recommended minimum value for this variable is 0x40 to avoid AE oscillation.
0xC874	cam_aet_ae_min_virt_dgain	[15:0]	Minimum value for the second digital gain that AE Track is permitted to use.
0xC876	cam_aet_ae_max_virt_dgain	[15:0]	Maximum value for the second digital gain that AE Track is permitted to use.
0xC878	cam_aet_ae_min_virt_again	[15:0]	Minimum value for the analog gain that AE Track is permitted to use.
0xC87A	cam_aet_ae_max_virt_again	[15:0]	Maximum value for the analog gain that AE Track is permitted to use.

Flicker Avoidance

Many CMOS sensors use a “rolling shutter” readout mechanism that greatly improves sensor data readout times. This allows pixel data to be read out much sooner than other methods that wait until the entire exposure is complete before reading out the first pixel data. The rolling shutter mechanism exposes a range of pixel rows at a time. This range of exposed pixels starts at the top of the image and then “rolls” down to the bottom during the exposure period of the frame. As each pixel row completes its exposure, it is ready to be read out. If the light source oscillates (flickers) during this rolling shutter exposure period, the image appears to have alternating light and dark horizontal bands.

If the sensor uses the traditional snapshot readout mechanism, in which all pixels are exposed at the same time and then the pixel data is read out, then the image may appear overexposed or underexposed due to light fluctuations from the flickering light source. Lights operating on AC electric systems produce light flickering at a frequency of 100 Hz or 120 Hz, twice the frequency of the power line.

To avoid this flicker effect, the exposure times must be multiples of the light source flicker periods. For example, in a scene lit by 120 Hz lighting, the available exposure times are 8.3 ms, 16.67 ms, 25 ms, 33.33 ms, and so on. (The need for an exposure time less than 8.3 ms under artificial light is extremely rare.)

The camera designer must first detect whether there is a flickering light source in the scene, and if so, determine its flickering frequency. In this case, the AE must limit the integration time to an integer multiple of the light’s flicker period.

In progressive mode, the control for the ASX340AT to switch between 50 and 60 Hz flicker avoidance is using the variable CAM_AET_FLICKER_FREQ_HZ (R0xC881). When the variable is set to 0x3 C 60 Hz flicker settings are being used and when it is set to 0x32 50 Hz flicker settings will be used.

In NTSC or PAL interlaced mode, use the following variables to set flicker avoidance:

- NTSC: NTSC_AET_FLICKER_FREQ_HZ (R0x9424)
- PAL: PAL_AET_FLICKER_FREQ_HZ (R0x9824)

The flicker steps sizes and number of flicker periods are all calculated for the user by the ASX340AT firmware.

Flicker Detection

The ASX340AT firmware has a flicker detection algorithm which has been optimized for use with NTSC and PAL frame rates (and progressive 50 and 60 fps). The algorithm is designed only to detect a 50 Hz or 60 Hz flicker source.

This algorithm is based on frame differences, and it is only able to search for and detect the opposite frequency to the one that it is currently configured to avoid. For example, if the ASX340AT is running NTSC with the flicker avoidance set to 60 Hz, the flicker detection will only search for a 50 Hz flicker source.

Table 26 shows the variables for configuring the flicker detection algorithm.

Table 26. VARIABLES FOR FICKER DETECTION

Address	Name	Bits	Function
0xC988[0]	cam_flicker_detect_fd_enable	[0]	Enable flicker detection 0: Disable flicker detection 1: Enable flicker detection Takes effect with a Refresh command
0xC988[1]	cam_flicker_detect_fd_auto_switch	[1]	Auto−switch flicker avoidance period control: 0: Auto switching disabled 1: Enable auto switching of the flicker period when a flicker source is detected in the scene. This notifies AE_Track to change the current flicker avoidance.
0xC000[4]	flicker_detect_fd_status_flicker_change_detected	[4]	Flicker detection status 0: No flicker has been detected. 1: Flicker detected in the current scene. Note: This flag is automatically cleared after a Change−Config, Refresh, or Standby operation.
0xC002[2]	flicker_detect_fd_mode_disable_fd_after_detection	[2]	If auto flicker detection is enabled, and this bit is set to 1, auto flicker detection will work only one time and then it will be disabled.
0xC00A	flicker_detect_total_num_attempts	[0:7]	Controls the number of attempts necessary for a complete validation test. Maximum is 16. An attempt is about 6 frames.
0xC00B	flicker_detect_successful_attempts_threshold	[0:7]	Controls the number of successful attempts within the last attempts (flicker_detect_total_num_attempts) to permit detection of flicker (greater−than or equal to)
0xC008	flicker_detect_current_status_attempts	[15:0]	Flicker detection status for the last 16 runs. It is essentially storing the last 16 results from flicker_detect_fd_status_flicker_change_detected, with the MSB representing the latest flicker detection status, and the LSB representing the least recent flicker detection status. The variable is automatically cleared after a Change−Config, Refresh, or Standby operation.

Manual Exposure

Manual exposure mode is available by disabling auto exposure. Users can manually adjust coarse integration time and fine integration time as well as analog gains for the desired exposure outcome.

There are two manual exposure types: full manual exposure mode and semi−manual exposure mode.

- Full manual exposure is enabled by setting ae_track_algo = 0. In this mode, the host directly changes the integration time and gains through the following control variables (these variables become R/W in manual exposure mode). See Table 27 for details.
  cam_sensor_control_coarse_integration_time
  cam_sensor_control_fine_integration_time
  cam_sensor_control_analog_gain
  cam_cpipe_control_dgain_second

  To avoid flicker, the integration time needs to be in multiples of flicker periods, and it is the responsibility of the host to choose flicker avoiding integration times.
- Semi−manual exposure is enabled by clearing ae_track_algo[1]. In this mode, the host directly configures ae_track_virt_exposure_log. Ae_track_virt_exposure_log is an unsigned fixed point variable in log2 format, specifying the desired number of pixel clocks as $2^{n.m}$, where n is the integer portion specified by the upper 8−bit, while m is the fractional portion specified by the lower 8−bit (unity = 256). For example, to specify the integration time as 1 flicker period for 60 Hz (1/120 = 8.333ms), assuming 27 MHz pixel clock, this equals to 225000 pixel clocks (27000000/120). Therefore, in semi−manual mode, ae_track_virt_exposure_log should be configured as 0x11C7 ($2^{17.779}$ = 225000). Based on ae_track_virt_exposure_log, the firmware will automatically update the integration times and gains to achieve the setup. Based on

ae_track_virt_exposure_log, the firmware will automatically update the integration times and gains to achieve the setup. The AE firmware will avoid flicker in semi−manual mode (when operating in zone 1).

The above key variables to be adjusted for full manual and semi−manual exposure modes are summarized in Table 27.

Table 27. VARIABLES USED FOR MANUAL EXPOSURE

Address	Name	Bits	Function
0xA804	ae_track_algo	[15:0]	0x0000: Full manual exposure (AE track disabled). Host controls the integration times and gains directly. 0x01FD: Semi−manual exposure (AE track enabled). Host controls exposure directly via ae_track_virt_exposure_log and FW updates the integration times and gains automatically. 0x01FF: Auto exposure mode
0xA806	ae_track_request_flag	[0]	ae_track_force_adjust_exposure: 1: Force to refresh the exposure settings in semi−manual exposure. Automatically cleared to 0 in auto exposure mode.
0xC840	cam_sensor_control_coarse_integration_time	[15:0]	Current coarse integration time specified in number of line_length_pck (i.e. the number of pixel clock periods in one line time). Read−only in auto exposure, and R/W in full manual exposure.
0xC842	cam_sensor_control_fine_integration_time	[15:0]	Current fine integration time specified in number of pixel clocks. Read−only in auto exposure, and R/W in full manual exposure.
0xC83A	cam_sensor_control_analog_gain	[15:0]	Applied 'virtual' global analog gain (unity = 32)
0xC84C	cam_cpipe_control_dgain_second	[15:0]	Applied second digital gain (unity = 128).
0xA828	ae_track_virt_exposure_log	[15:0]	Indicates the current virtual exposure in log2. Read−only in auto exposure, and R/W in semi−manual exposure mode.

COLOR TUNING

This section discusses the color tuning functions: White Balance (WB), Color Correction Matrix (CCM), color saturation, Auto White Balance (AWB), and manual white balance. It also covers the procedure for generating AWB and CCM settings using SensorTune tool.

White Balance

Color temperature is a way of measuring the characteristic of a light source. It is based on the ratio of the amount of blue light component to the amount of red light component, and the green light component is ignored. A white object may not appear the same "white" under lights with different color temperatures. White balance is the process of removing unrealistic color casts, so that objects which appear white to the human eye are rendered white in the image. White balance is the first step for achieving desired color rendition results.

In an ideal situation, each gray grid in a Macbeth color chart should have red, green, and blue color components at the same level for white balance. In Figure 41, the cross−section of gray row shows red, green, and blue at similar intensity level for white balance.

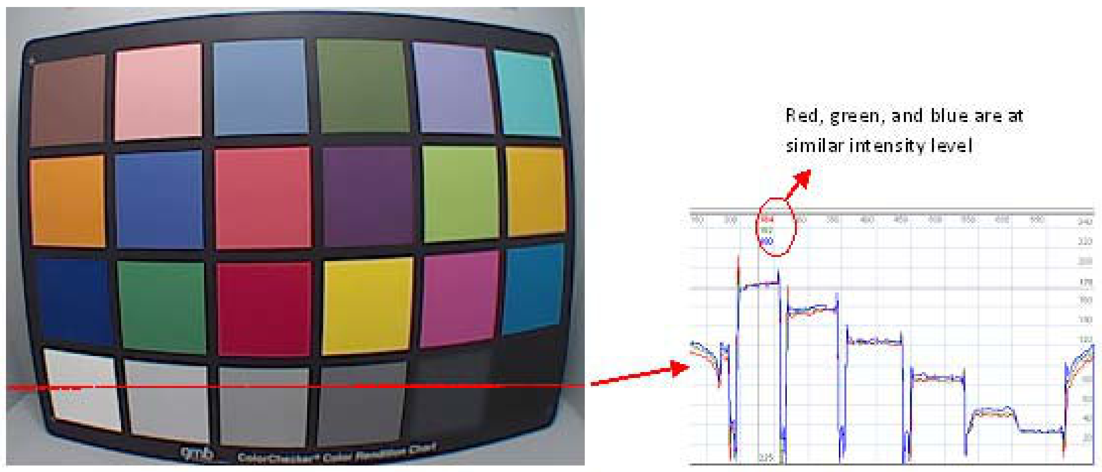


Figure 41. White Balance Graph

Color Correction Matrix

To achieve good color rendition and color saturation, interpolated colors of all pixels are subjected to color correction. The color correction is a linear transformation of the image with a 3x3 color correction matrix.

The optimal values of the correction matrix elements depend on the spectrum of light incident on the sensor. They can be either programmed by the user or automatically selected by the auto white balance algorithm.

The color correction matrix consists of nine values, each of which represents a digital gain factor on the corresponding color channel with the diagonal elements representing the gain factors on each color channel and the off diagonal terms representing the gain factors to compensate for color crosstalk. The matrix is normalized so that the sum of each row is "1." All the color correction matrix values are stored in the CCM variable map.

Table 28. COLOR CORRECTION MATRIX STRUCTURE

Saturation Type/Gain	Gain R	Gain G	Gain B
Saturation Red	ccm_0	ccm_1	ccm_2
Saturation Green	ccm_3	ccm_4	ccm_5
Saturation Blue	ccm_6	ccm_7	ccm_8

The ASX340AT supports three color correction matrices: one is for the red−rich illumination (normally 2856 K color temperature and named left matrix), the middle one is aimed for fluorescent light (normally 3850 K color temperature), and the third one is for blue−rich illumination (normally would be 6500 K color temperature and named right matrix).

All matrices have associated R/G and B/G ratios (cam_awb_ccm_[l/m/r]_[r/b]g_gain) describing the gain needed for the red and blue channels for those particular color temperatures. Also, each CCM has a color temperature associated with it (cam_awb_ccm_[l/m/r]_ctemp).

Using the color temperature (cam_awb_color_temperat ure), the left and middle or middle and right matrices are mixed using linear interpolation to generate the CCM for the next frame.

The interpolated RGB values are transformed by the color correction matrix (CCM) into color−corrected R', G', and B' values. The color correction matrix is uploaded by the

AWB firmware driver into the corresponding registers in the color pipeline when AWB has settled and the White Balance has adjusted.

Auto White Balance

AWB algorithm is designed to compensate for the effects of changing spectra of the scene illumination on the quality of the color rendition. The algorithm consists of two major parts:

- A measurement engine performing statistical analysis of the image.
- AWB driver calculating the digital and sensor core analog gains, determining the color temperature, and performing the selection of the optimal color correction matrix.

The AWB driver analyzes measurement engine data and sets appropriate digital AWB gains and current color temperature. The color temperature then determines the color correction matrix position, which defines the current matrix coefficients and digital gain ratio.

Table 29. VARIABLES FOR SETTING AWB GAINS

Address	Name	Function
R0xAC12	awb_r_gain	Current red channel gain
R0xAC14	awb_b_gain	Current blue channel gain
R0xC8BE	cam_awb_ccm_l_rg_gain	R/G gain ratio for Left Matrix
R0xC8C0	cam_awb_ccm_l_bg_gain	B/G gain ratio for Left Matrix
R0xC8C2	cam_awb_ccm_m_rg_gain	R/G gain ratio for Middle Matrix
R0xC8C4	cam_awb_ccm_m_rg_gain	B/G gain ratio for Middle Matrix
R0xC8C6	cam_awb_ccm_r_rg_gain	R/G gain ratio for Right Matrix
R0xC8C8	cam_awb_ccm_r_rg_gain	R/G gain ratio for Right Matrix

How to Speed Up or Slow Down AWB Response

It is possible to speed up or slow down the AWB response.

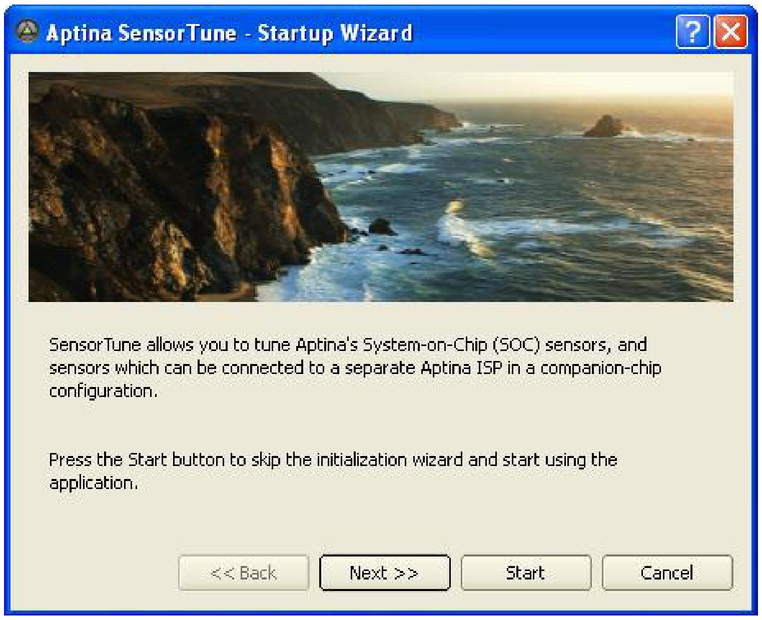
1. Set awb_enable_pre_awb_ratios_damping = 1 (R0xAC02[6] = 0x01)
2. Set awb_pre_awb_ratios_tracking_speed (R0xAC16): Speed of AWB digital gains buffering (32 = fastest, 1 = slowest).

NOTE: Setting the AWB damping too fast may cause AWB to be unstable.

Generating AWB and CCM Using Sensor Tune

This section explains how to use the SensorTune tool to produce AWB and CCM settings for ASX340AT. The procedure is as follows:

1. Go to *Start* → *All Programs* → **onsemi** *Imaging* → *Tools* → *Sensor Tune (AWB and CCM)*, and click on *Next* >>.

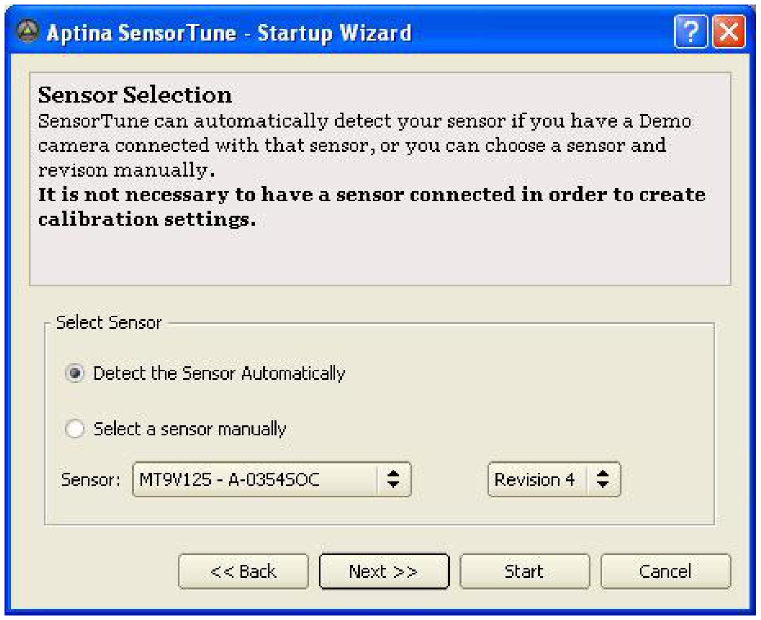


SensorTune allows you to tune Aptina's System-on-Chip (SOC) sensors, and sensors which can be connected to a separate Aptina ISP in a companion-chip configuration.

Press the Start button to skip the initialization wizard and start using the application.

Figure 42.

2. Choose either Detect the Sensor Automatically or Select a sensor manually



Sensor Selection

SensorTune can automatically detect your sensor if you have a Demo camera connected with that sensor, or you can choose a sensor and revison manually.

It is not necessary to have a sensor connected in order to create calibration settings.

Select Sensor

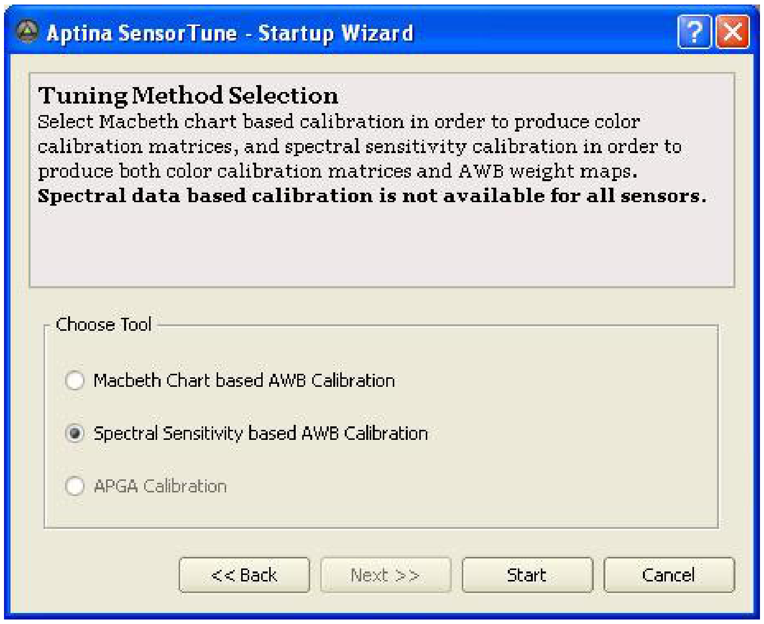
[✓ Detect the Sensor Automatically]

Select a sensor manually

Sensor: MT9V125 - A-0354SOC   Revision 4

Figure 43.

3. Select Spectral Sensitivity based AWB Calibration, and then click Start.



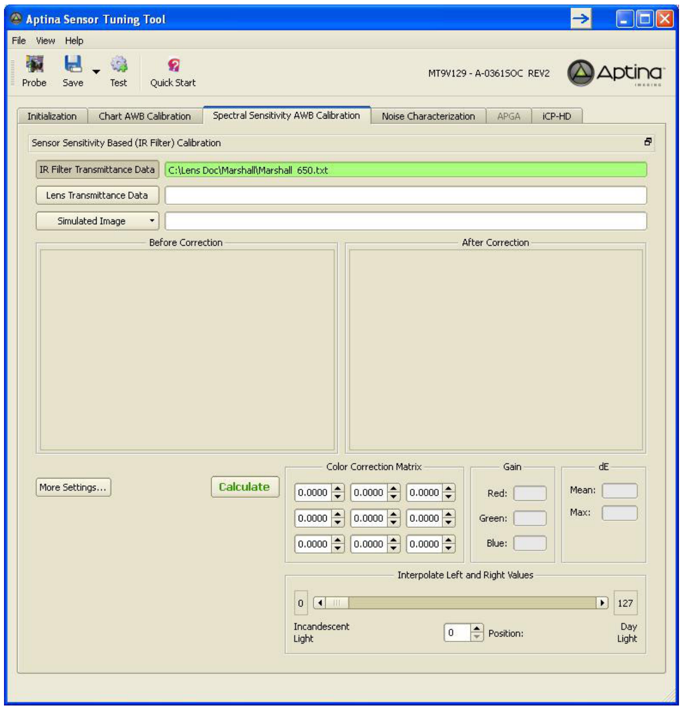
Tuning Method Selection
Select Macbeth chart based calibration in order to produce color calibration matrices, and spectral sensitivity calibration in order to produce both color calibration matrices and AWB weight maps.
Spectral data based calibration is not available for all sensors.

Choose Tool
- ( ) Macbeth Chart based AWB Calibration
- [✓ Spectral Sensitivity based AWB Calibration]
- ( ) APGA Calibration

Figure 44.

4. Load the IR Filter Transmittance Data (mandatory) and Lens Transmittance Data (optional). The IR filter data is typically saved in a text file. There are two columns in the data:

- The first column is the wavelength in nm.
- The second column is the band pass filter value (normalized or in percentage) at the corresponding wavelength.



MT9V129 - A-0361SOC REV2

Sensor Sensitivity Based (IR Filter) Calibration

IR Filter Transmittance Data: C:\Lens Doc\Marshall\Marshall 650.txt
Lens Transmittance Data:
Simulated Image:

Color Correction Matrix:

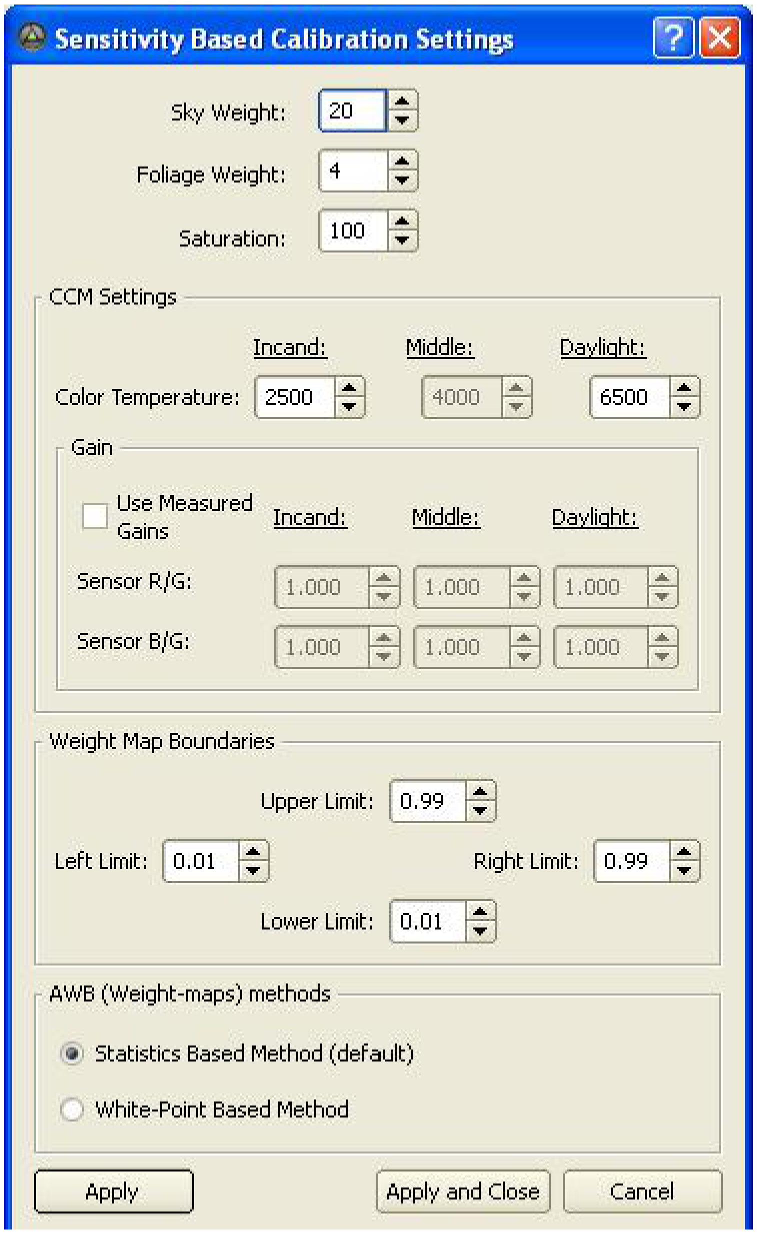
0.0000	0.0000	0.0000
0.0000	0.0000	0.0000
0.0000	0.0000	0.0000

Gain — Red: Green: Blue:
dE — Mean: Max:

0 … 127
Position: 0

Figure 45.

5. Click on *More Settings* to bring up a new menu.



Sky Weight: 20
Foliage Weight: 4
Saturation: 100

CCM Settings

	Incand:	Middle:	Daylight:
Color Temperature:	2500	4000	6500

Gain

☐ Use Measured Gains

	Incand:	Middle:	Daylight:
Sensor R/G:	1.000	1.000	1.000
Sensor B/G:	1.000	1.000	1.000

Weight Map Boundaries

Upper Limit: 0.99
Left Limit: 0.01
Right Limit: 0.99
Lower Limit: 0.01

AWB (Weight-maps) methods

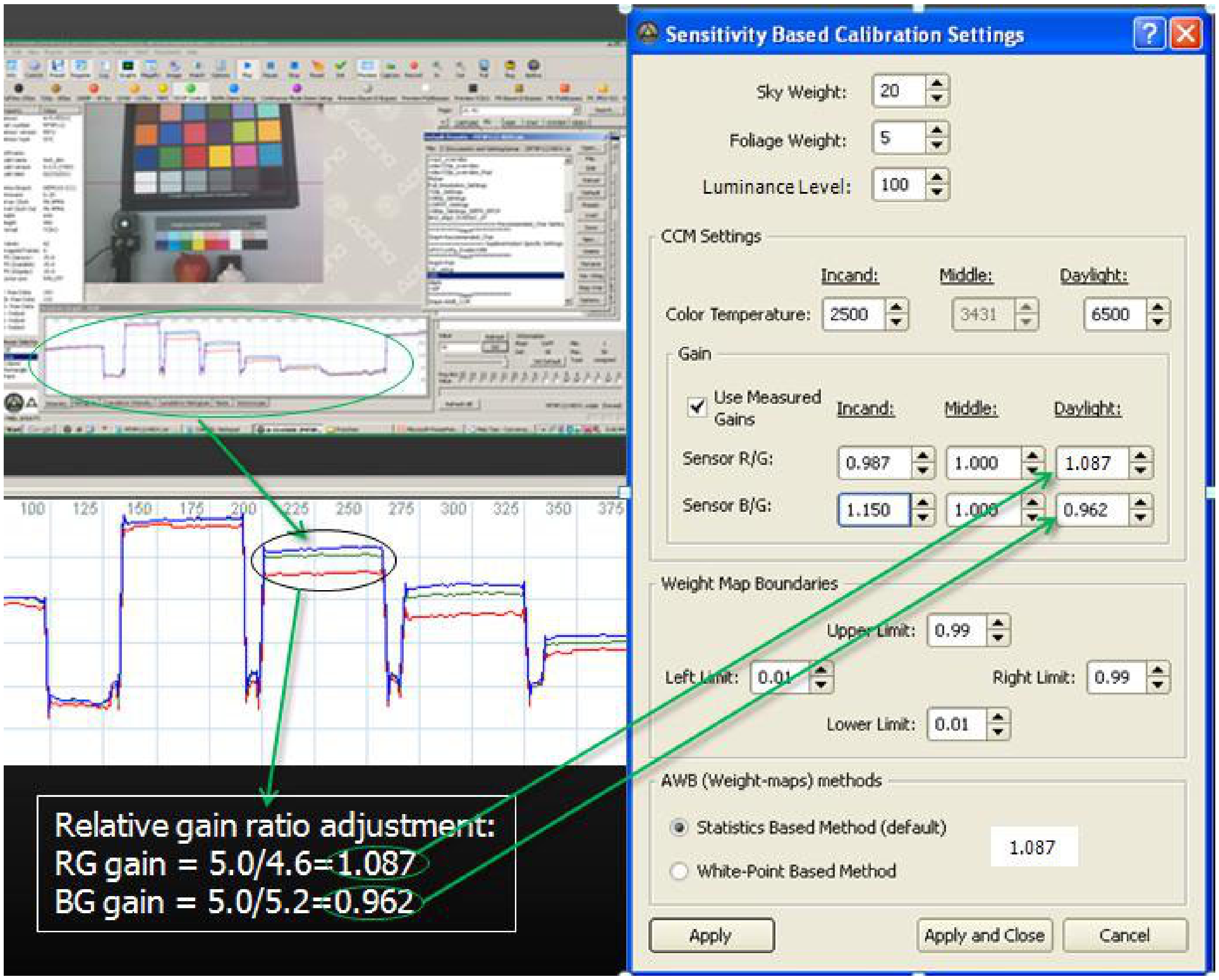
● Statistics Based Method (default)
○ White-Point Based Method

Figure 46.

- Sky Weight
  If this value is increased, the weighting giving to "blue sky"
- Foliage Weight
  As this value is increased, it indicates that more "foliage" is in the scene.
- Color Temperature
  Input the color temperature used for Incandescent and Daylight.
- Measured Gains
  Selecting the measured gains allows the user to input the relative gains for improving white balance. A first round of AWB/CCM calculation should be performed with default settings.

As shown in Figure 47, at D65 (daylight), the cross−section graph of gray grids shows that the red gain is low while the blue gain is high for white balance. To achieve the white balance, the red gain needs to be increased, while the blue gain needs to be decreased. By calculating the ratio of intensity in red channel over green channel, as shown in Figure 39, the relative gain for red channel is proposed to increase to 1.087. Similarly, the relative gain for blue channel is proposed to decrease to 0.962 (assuming the gain of green channel keeps the same). Input these relative gain values to Sensor Tune under Daylight column as shown in Figure 39, and re−calculate the AWB/CCM calibration. Acquire relative gains at all three color temperatures if needed for AWB/CCM recalculation.

It is worth noting that changing ratio gains at one color temperature may also impact the white balance performance at other color temperatures since ratio gain modifications will change the whole AWB weighted map and color correction interpolation. Therefore, this procedure might need to be repeated a couple of rounds to achieve desired white balance under all lighting conditions.



Relative gain ratio adjustment:
RG gain = 5.0/4.6=1.087
BG gain = 5.0/5.2=0.962

Sensitivity Based Calibration Settings

Sky Weight: 20
Foliage Weight: 5
Luminance Level: 100

CCM Settings

	Incand:	Middle:	Daylight:
Color Temperature:	2500	3431	6500

Gain

[✓] Use Measured Gains	Incand:	Middle:	Daylight:
Sensor R/G:	0.987	1.000	1.087
Sensor B/G:	1.150	1.000	0.962

Weight Map Boundaries

Upper Limit: 0.99
Left Limit: 0.01
Right Limit: 0.99
Lower Limit: 0.01

AWB (Weight-maps) methods

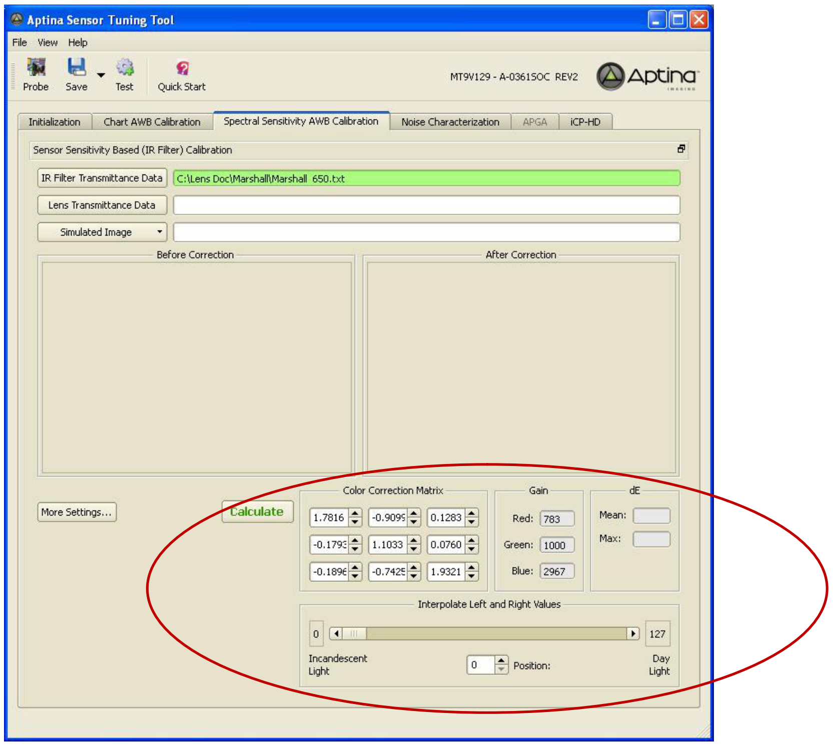
[✓ Statistics Based Method (default)]
White-Point Based Method

1.087

Figure 47. Input Relative Gain Ratio for Sensor Tune

- Weight Map Boundaries
  These limits determine how tightly the weight table boundaries are with respect to the white points.
- AWB Weight Map Method
  Using the Statistics−based method will account for part−to−part variation.

6. Click on **Calculate** to generate the AWB and CCM matrices



IR Filter Transmittance Data: C:\Lens Doc\Marshall\Marshall 650.txt

Color Correction Matrix:

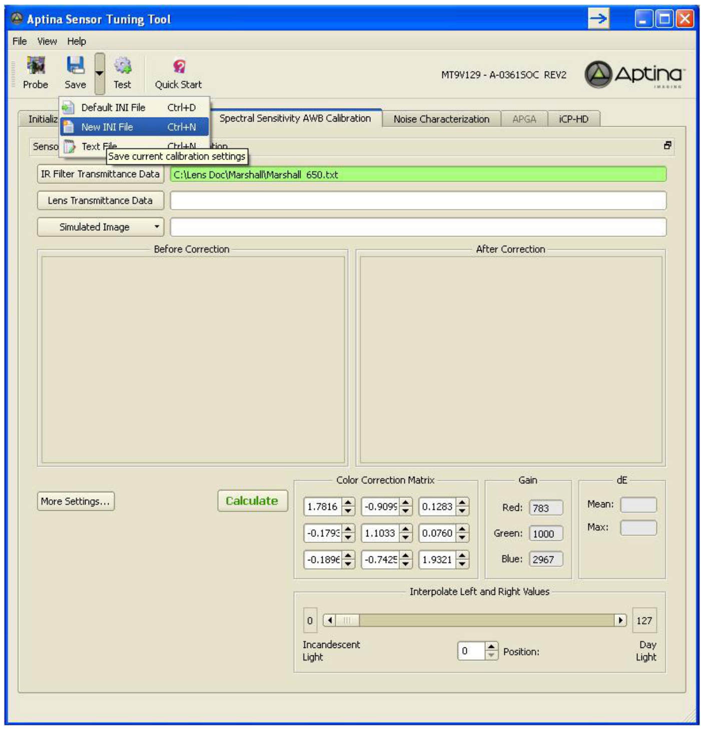
1.7816	-0.9099	0.1283
-0.1793	1.1033	0.0760
-0.1896	-0.7425	1.9321

Gain: Red: 783, Green: 1000, Blue: 2967

Interpolate Left and Right Values: 0 – 127, Position: 0

Figure 48.

7. Save the settings to an ini file.



IR Filter Transmittance Data: C:\Lens Doc\Marshall\Marshall 650.txt

Color Correction Matrix:

1.7816	-0.9099	0.1283
-0.1793	1.1033	0.0760
-0.1896	-0.7425	1.9321

Gain: Red: 783, Green: 1000, Blue: 2967

Interpolate Left and Right Values: 0 – 127, Position: 0

Figure 49.

8. Validate the ini file by applying it to the demo camera or module and ensure that images are giving expected results.

How to Produce Color−Tinted Effects

In the AWB driver, there are several variables that will produce global color offsets (tints) to the images. These variables operate on the color correction matrix as a normalization coefficient. Some color tints can be applied to the output of the CCM. However instead of using a new gain module the CCM itself is updated to get extra gain for the Red, Green and Blue channels based on the color tints setup.

There are two sets of tints: red−light tint matrix (i.e. cam_awb_k_r_l, cam_awb_k_g_l, and cam_awb_k_b_l) and blue−light tint matrix (i.e. cam_awb_k_r_r, cam_awb_k_g_r, and cam_awb_k_b_r). Table 30 includes the color tint related variables.

Table 30. COLOR−TINT: VARIABLES FOR COLOR−TINTED EFFECTS

Address	Name	Function
R0xC8CA	cam_awb_ccm_l_ctemp	Color temperature for the Left Matrix (in Kelvin)
R0xC8E6	cam_awb_color_temperature	Current matrix color temperature (in Kelvin)
R0xC902	cam_awb_tints_ctemp_threshold	Color temperature threshold for color tint application.
R0xC904	cam_awb_k_r_l	Controls the tint for the red channel at the color temperature set by cam_awb_ccm_l_ctemp
R0xC905	cam_awb_k_g_l	Controls the tint for the green channel at the color tempera-ture set by cam_awb_ccm_l_ctemp
R0xC906	cam_awb_k_b_l	Controls the tint for the blue channel at the color temperature set by cam_awb_ccm_l_ctemp
R0xC907	cam_awb_k_r_r	Controls the tint for the red channel at the color temperature set by cam_awb_tints_ctemp_threshold
R0xC908	cam_awb_k_g_r	Controls the tint for the green channel at the color tempera-ture set by cam_awb_tints_ctemp_threshold
R0xC909	cam_awb_k_b_r	Controls the tint for the blue channel at the color temperature set by cam_awb_tints_ctemp_threshold

The functionality is summarized below:

- When cam_awb_color_temperature is equal to cam_awb_ccm_l_ctemp, only the red−light tint matrix has effects;
- When cam_awb_color_temperature is larger than cam_awb_ccm_l_ctemp, but smaller than cam_awb_tints_ctemp_threshold, both red−light and blue−light tint matrices are used for interpolation to generate the current color tint effect;
- When cam_awb_color_temperature is larger than cam_awb_tints_ctemp_threshold, only the blue−light tint matrix has effects.

Manual White Balance

The ASX340AT allows manual white balance by clearing cam_awb_mode_auto (R0xC8FF[1]), which controls the switch between auto and manual white balance modes. In manual WB mode, R0xC8E6, i.e. cam_awb_color_temperature needs to be manually set to the current color temperature. The following presets allow to switch between manual white balance and auto white balance.

```
[Enable Manual WB − A Light]
REG = 0xC8FF, 1
REG = 0xC8E6, 2800
[Enable Manual WB − D65]
REG = 0xC8FF, 1
REG = 0xC8E6, 6500
[Enable Manual WB − CWF]
REG = 0xC8FF, 1
REG = 0xC8E6, 4400
[Enable Auto WB]
REG = 0xC8FF, 3
```

Note that *cam_awb_color_temperature* is read−only in AWB, and it has read/write permission in manual WB mode.

Table 31. VARIABLES FOR MANUAL WHITE BALANCE

Address	Name	Function
R0xC8FF	cam_awb_awbmode	Bit[1]: cam_awb_mode_auto 0: Manual mode − the user must manually change the color temperature (cam_awb_color_temperature); the gain ratios are then adjusted accordingly. 1: Auto mode − AWB calculates the gain ratios and color temperature.
R0xC8E6	cam_awb_color_temperature	Current matrix color temperature. Read−only in AWB mode; needs to be enabled with writing permission for manual white balance mode.

GAMMA CORRECTION

Gamma Curve Selection

The ASX340AT includes a block for gamma correction that can adjust its shape based on brightness to enhance the performance under certain lighting conditions. Two custom gamma correction tables may be uploaded: one corresponding to a brighter lighting condition (first curve), the other corresponding to a darker lighting condition (second curve) as shown in Figure 50.

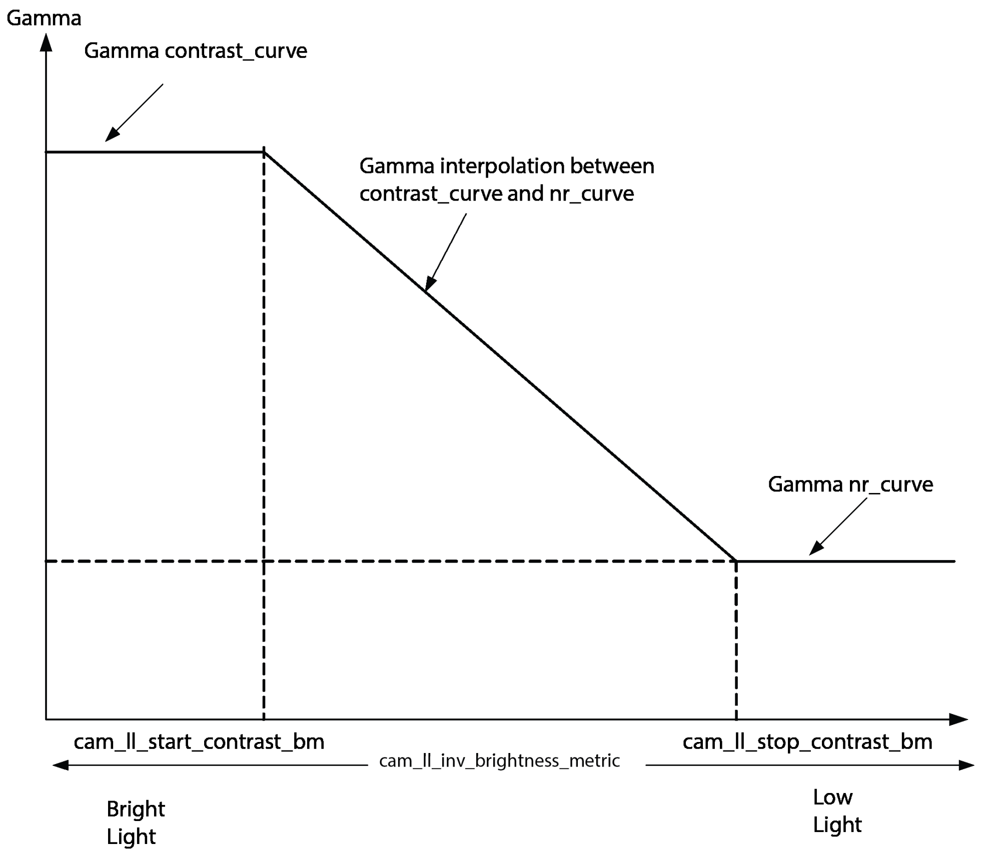


Figure 50. Gamma

When the lighting condition is in between, the interpolated curve is used for gamma correction.

Table 32. GAMMA CURVE SELECTION

Address	Name	Bits	Function
0xBC07	ll_gamma_select	[7:0]	0x00: Auto curve select based on settings of cam_ll_start_contrast_bm and cam_ll_stop_contrast_bm 0x01: Contrast curve is only used 0x02: Noise reduction curve is only used
0xC93E	cam_ll_start_contrast_bm	[15:0]	In auto curve select mode, when cam_ll_inv_bright-ness_metric is below this value, contrast curve (first curve) is used for gamma correction.
0xC940	cam_ll_stop_contras_bm	[15:0]	In auto curve select mode, when cam_ll_inv_bright-ness_metric is above this value, noise reduction curve (second curve) is used for gamma correction.
0xC958	cam_ll_inv_brightness_metric	[15:0]	A measure of the scene brightness (the lower the value, the brighter the scene).

Contrast Curve and Noise Reduction Curve

For each gamma curve, there are 19 knee points. The 19 knee points can be generated automatically based on a small number of variables, or be customized by users. *Cam_ll_llmode* variable controls whether the auto−calculated or customer−defined curve will be used. Table 33 shows the variables for generating contrast and noise reduction gamma curves.

When auto−calculation is selected for gamma curves, Figure 51 shows the concept of how the variables *cam_ll_XX_contrast_gradient* and *cam_ll_XX_contrast_ luma_percentage* interact to produce a curve. To generate a linear gamma curve with auto−calculation by firmware, use the following settings:

- cam_ll_gamma = 0x0064 (i.e., 1.0)
- cam_ll_start_contrast_gradient = 0x20 (i.e., 1.0)
- cam_ll_stop_contrast_gradient = 0x20 (i.e., 1.0)

On *Control → Gamma → Manual Adjust* page, DevWare provides a way of calculating gamma curves as well. To use gamma curves generated by this page, set cam_ll_llmode = 0. Changing sliders on this page directly writes to the 19 knee points of corresponding gamma curves. To create a linear gamma curve using this method, use the following settings:

- Gamma slider sets to 1
- Black correct slider sets to 0
- Contrast slider sets to 1

Table 33. CONTRAST CURVE AND NOISE REDUCTION CURVE GENERATION

Address	Name	Bits	Function
0xC926	cam_ll_llmode	[1:0]	0x00: User will program 19 knee point gamma curves. 0x01: Firmware will calculate 19 knee points for contrast curve (first curve or table). 0x02: Firmware will calculate 19 knee points for noise reduction curve (second curve or table). 0x03: Firmware will calculate both contrast and noise reduction curves.
0xC942	cam_ll_gamma	[15:0]	The value of the gamma curve (in unit of 100). This applies to both gamma curves.
0xC944	cam_ll_start_contrast_gradient	[15:0]	The slope value of the first curve at inflection point
0xC945	cam_ll_start_contrast_luma_percentage	[15:0]	The slope value of the second curve at inflection point
0xC946	cam_ll_stop_contrast_luma_percentage	[7:0]	The percentage of target luma for the inflection point in the first curve.
0xC947	cam_ll_stop_contrast_luma_percentage	[7:0]	The percentage of target luma for the inflection point in the second curve.
0xBC0A 0xBC0B 0 0 0 0xBC1C	ll_gamma_contrast_curve_0 ll_gamma_contrast_curve_1 0 0 0 ll_gamma_contrast_curve_18	[7:0]	Customer defined knee points for contrast curve. The X coordinate of each point is fixed individually. They are as follows (in the order of the 19 knee points): 0, 4, 8, 16, 32, 48, 64, 80, 96, 112, 128, 144, 160, 176, 192, 208, 224, 240, 256.
0xBC1D 0xBC1E 0 0 0 0xBC2F	ll_gamma_nrcurve_0 ll_gamma_nrcurve_1 0 0 0 ll_gamma_nrcurve_18	[7:0]	Customer defined knee points for noise reduction curve. The X coordinate of each point is fixed individually. They are as follows (in the order of the 19 knee points): 0, 4, 8, 16, 32, 48, 64, 80, 96, 112, 128, 144, 160, 176, 192, 208, 224, 240, 256.

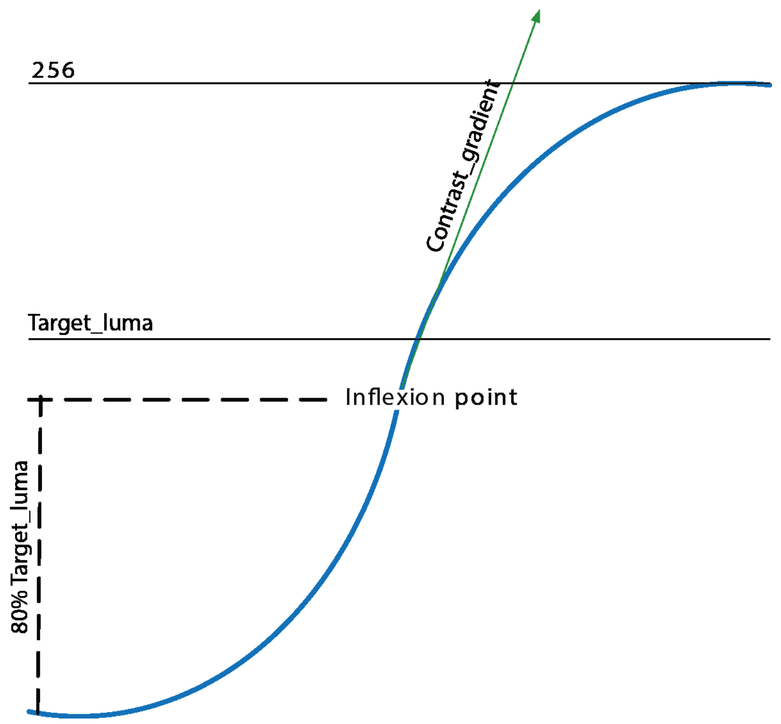


Figure 51. Example of Auto−Generated Curve

Fade−to−Black

The final stage of the gamma flow is the enabling and use of Fade−to−Black. The ASX340AT IFP allows for the image to fade to black under extreme low−light conditions. This feature enables users to optimize the performance of the sensor under low−light conditions. It minimizes the perception of noise and artifacts while the available illumination is diminishing.

This feature has two user−set points that reference the brightness of the scene. When the Fade−to−Black starts, it will interpolate to the end point as the light falls until it gets to the end point. When at and after the end point, the image will be black.

Table 34. VARIABLES USED FOR FADE−TO−BLACK

Address	Name	Bits	Function
0xBC02	ll_mode	[3]	0x00: User will program 19 knee point gamma curves. 0x01: Firmware will calculate 19 knee points for contrast curve (first curve or table). 0x02: Firmware will calculate 19 knee points for noise reduction curve (second curve or table). 0x03: Firmware will calculate both contrast and noise reduction curves.
0xBC3A	ll_average_luma_fade_to_black	[15:0]	The value of the gamma curve (in unit of 100). This applies to both gamma curves.
0xC94C	cam_ll_start_fade_to_black_luma	[15:0]	The slope value of the first curve at inflection point
0xC94E	cam_ll_stop_fade_to_black_luma	[15:0]	The slope value of the second curve at inflection point

onsemi recommends setting cam_ll_start_fade_to_black_luma below 0.5 lux and cam_ll_stop_fade_to_black_luma at around 0.1 lux. However, due to the low light sensitivity of the ASX340AT it is at the discretion of the user.

LOW LIGHT TUNING

In the ASX340AT there are a number of different image enhancements based on brightness metric (i.e. *cam_ll_start_brightness* and *cam_ll_stop_brightness*) and gain metric (i.e. *cam_ll_start_gain_metric* and *cam_ll_stop_gain_metric*). Two different modules in the color pipeline of the ASX340AT affect the sharpness of the image: noise reduction, and aperture correction (which is part of the demosaicing algorithm). While noise reduction will blur the image, aperture correction applies a sharpening filter with adjustable strength to regain sharpness after color interpolation. This sharpening filter also includes a threshold to avoid amplifying noise.

It is also possible to increase or decrease the amount of color saturation and de−saturation. The amount of saturation, aperture correction, and noise reduction are dynamically controlled by the firmware based on gain settings and lighting conditions.

Table 35. VARIABLES USED FOR FADE−TO−BLACK

Address	Name	Bits	Function
R0xC928	cam_ll_start_brightness	[15:0]	Start point of brightness level for LL settings.
R0xC92A	cam_ll_stop_brightness	[15:0]	Stop point of brightness level for LL settings. The value should be larger than cam_ll_start_brightness.
R0xC92C	cam_ll_start_saturation	[7:0]	Start value for saturation.
R0xC92D	cam_ll_end_saturation	[7:0]	End value for saturation.
R0xC92E	cam_ll_start_desaturation	[7:0]	Start value for desaturation.
R0xC92F	cam_ll_end_desaturation	[7:0]	Start value for desaturation.
R0xC930	cam_ll_start_demosaic	[[7:0]	Start value for interpolation.
R0xC931	cam_ll_start_ap_gain	[7:0]	Start value for aperture correction.
R0xC932	cam_ll_start_ap_thresh	[7:0]	Start value for aperture threshold.
R0xC933	cam_ll_stop_demosaic	[7:0]	Stop value for interpolation.
R0xC934	cam_ll_stop_ap_gain	[7:0]	Stop value for aperture correction.
R0xC935	cam_ll_stop_ap_thresh	[7:0]	Stop value for aperture threshold.
R0xC936	cam_ll_start_nr_red	[7:0]	Start value for the red channel noise reduction thresh-old. If the delta between a surround pixel and the center pixel is within this threshold, the surround pixel will be used to calculate the average to replace the current pixel value. (same rule applies to green and blue thresholds below)
R0xC937	cam_ll_start_nr_green	[7:0]	Start value for the green channel noise reduction threshold.
R0xC938	cam_ll_start_nr_blue	[7:0]	Start value for the blue channel noise reduction thresh-old
R0xC939	cam_ll_start_nr_thresh	[7:0]	Start value for the max−min threshold for enabling noise reduction. If the delta between the max and min values of surrounding pixels is less than this threshold, noise reduction is enabled. This globally controls en-abling/disabling of red, green, and blue channels.
R0xC93A	cam_ll_stop_nr_red	[7:0]	Stop value for the red channel noise reduction thresh-old
R0xC93B	cam_ll_stop_nr_green	[7:0]	Stop value for the green channel noise reduction threshold
R0xC93C	cam_ll_stop_nr_blue	[7:0]	Stop value for the blue channel noise reduction thresh-old
R0xC93D	cam_ll_stop_nr_thresh	[7:0]	Stop value of gain metric for noise reduction function at darker condition. This value should be larger than cam_ll_start_gain_metric
R0xC948	cam_ll_start_gain_metric	[15:0]	Start value of gain metric for noise reduction function at brighter condition.
R0xC94A	cam_ll_stop_gain_metric	[15:0]	Stop value of gain metric for noise reduction function at darker condition. This value should be larger than cam_ll_start_gain_metric

Table 35. VARIABLES USED FOR FADE−TO−BLACK(CONTINUED)

Address	Name	Bits	Function
R0xC958	cam_ll_inv_brightness_metric	[15:0]	A measure of the scene brightness (the lower the value, the brighter the scene).
R0xC95A	cam_ll_gain_metric	[15:0]	A measure of the effective gain applied in the system.

Table 35 lists the variables for low light image enhancements. For saturation, demosaic, and aperture parameters, they depend on the brightness metric, i.e. cam_ll_inv_brightness_metric to set the bright light and low light thresholds. For noise reduction parameters, however, they depend on the gain metric, i.e. cam_ll_gain_metric.

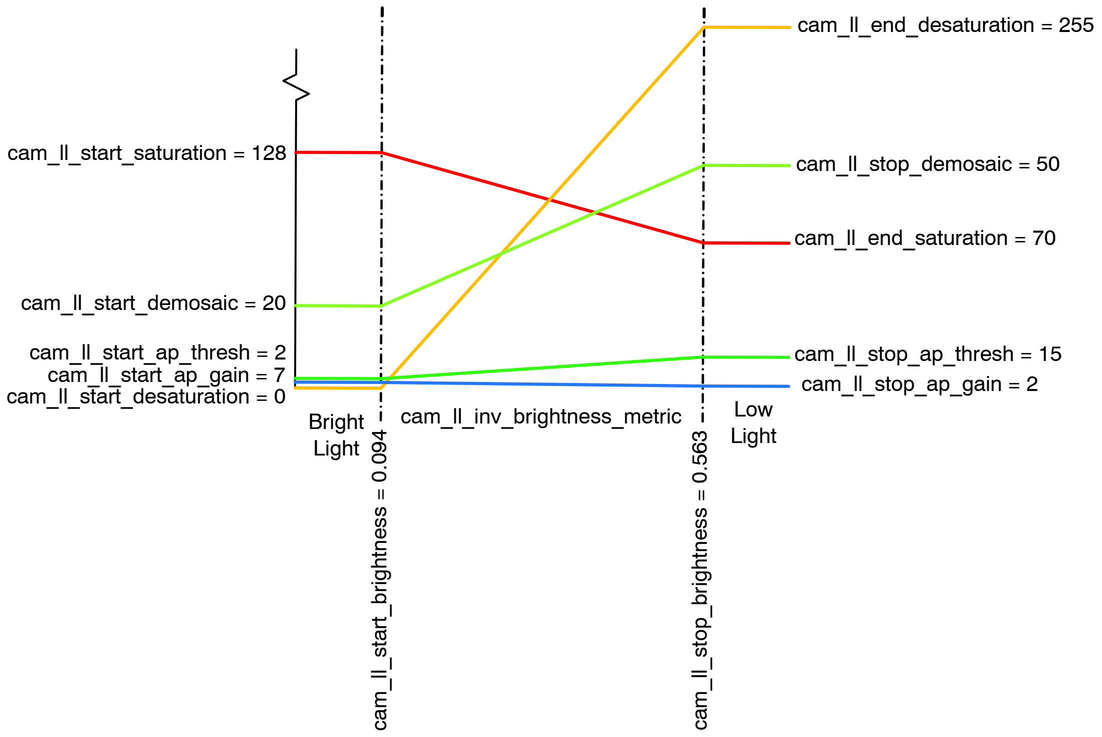


Figure 52. Brightness Metric

As shown in Figure 52, the low light tuning of saturation, demosaic, and aperture functions according to the following:

- When cam_ll_inv_brightness_metric is smaller than cam_ll_start_brightness, the start parameter will be used for each corresponding threshold;
- When cam_ll_inv_brightness_metric is between cam_ll_start_brightness and cam_ll_stop_brightness, the linearly interpolated value between the start and stop parameters will be used for each corresponding threshold;
- When cam_ll_inv_brightness_metric is larger than cam_ll_stop_brightness, the stop parameter will be used for each corresponding threshold.

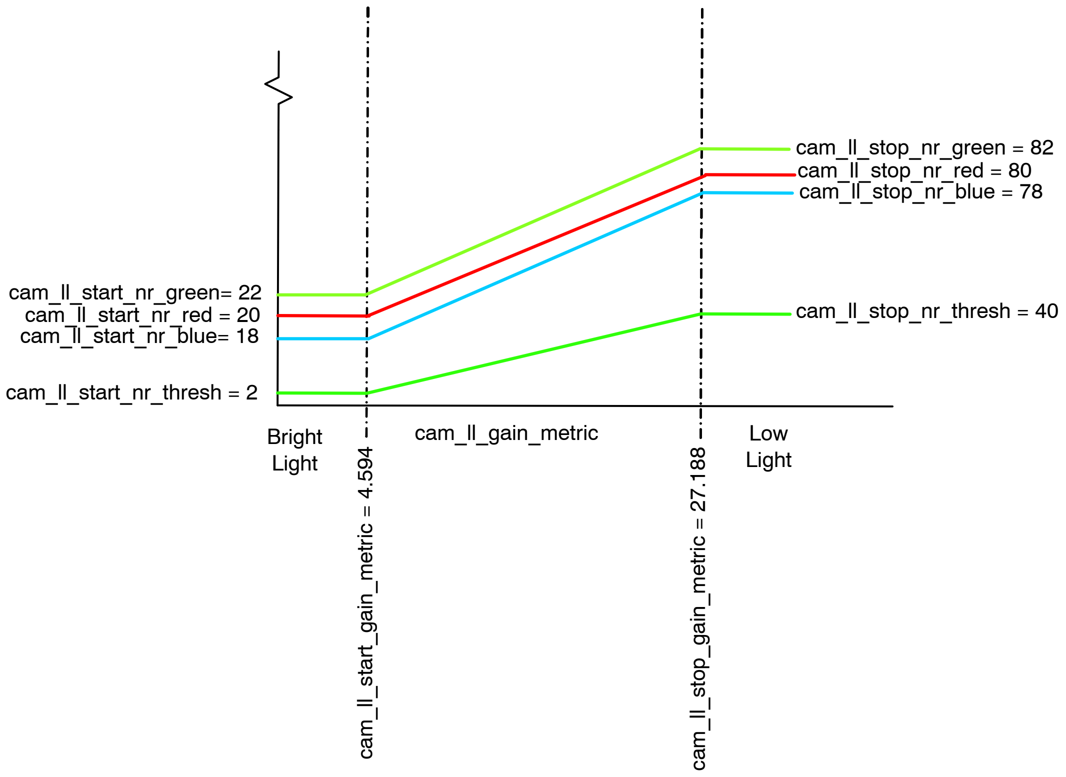


Figure 53. Noise Reduction with Gain Metric

As shown in Figure 53, the low light noise reduction settings are based on the following:

- When cam_ll_gain_metric is smaller than cam_ll_start_gain_metric, the start parameter will be used for each corresponding threshold;
- When cam_ll_gain_metric is between cam_ll_start_gain_metric and cam_ll_stop_gain_metric, the linearly interpolated value between the start and stop parameters will be used for each corresponding threshold;
- When cam_ll_gain_metric is larger than cam_ll_stop_gain_metric, the stop parameter will be used for each corresponding threshold.

How to Set Brightness Metric

To set up the *cam_ll_start_brightness* and *cam_ll_stop_brightness* points:

1. Set camera up for desired frame rate and with 100 lux light (The light level is only a guide).
2. Note the value of *cam_ll_inv_brightness_metric*; store this value in *cam_ll_start_brightness.*
3. Change light level to 20 lux (the light level is only a guide).
4. Note the value of *cam_ ll_inv_brightness_metric*; store this value in *cam_ll_stop_brightness.*

Effects of Low Light Variables on Image Quality

1. Saturation (cam_ll_start_saturation/cam_ll_end_saturation) This variable will be globally applied to the CCM and should be used as a fine tuning, after CCM has been tuned. Typically, **onsemi** recommends that cam_ll_end_saturation be 30% lower than cam_ll_start_saturation, but it is left to the discretion of the user to tune the part as they desire. See Figure 54 for images that show the extremes of setting the variable.

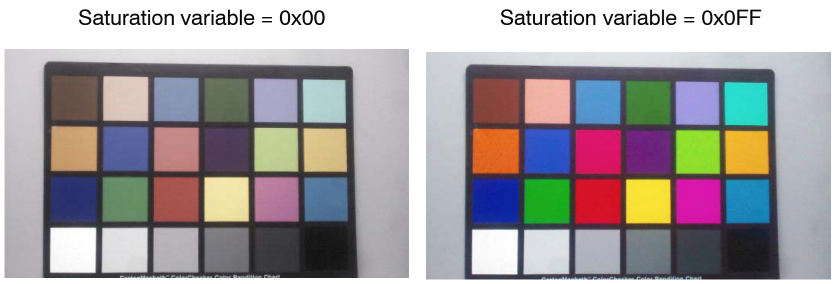


Figure 54. Saturation Variable Min and Max

2. Desaturation (cam_ll_start_desaturation, cam_ll_end_desaturation)
   This variable will be globally applied to the CCM and will desaturate the image. The effect is quite subtle. Setting the variable to 0x00 gives the minimum amount of desaturation. Setting the variable to 0xFF gives the maximum desaturation. But this is typically left as default. Use saturation tuning instead.
3. Demosaic (cam_ll_start_demosaic, cam_ll_stop_demosaic)
   This is the threshold for which demosaic edge detection will occur. Setting this variable will cause blurring of the image and also edge artifacts. Cam_ll_stop_demosaic must be set to a higher value than cam_ll_start_demosaic as when in low light a level of blurring is acceptable, but in balance with not having obvious edge artifacts.(See Figure 55).

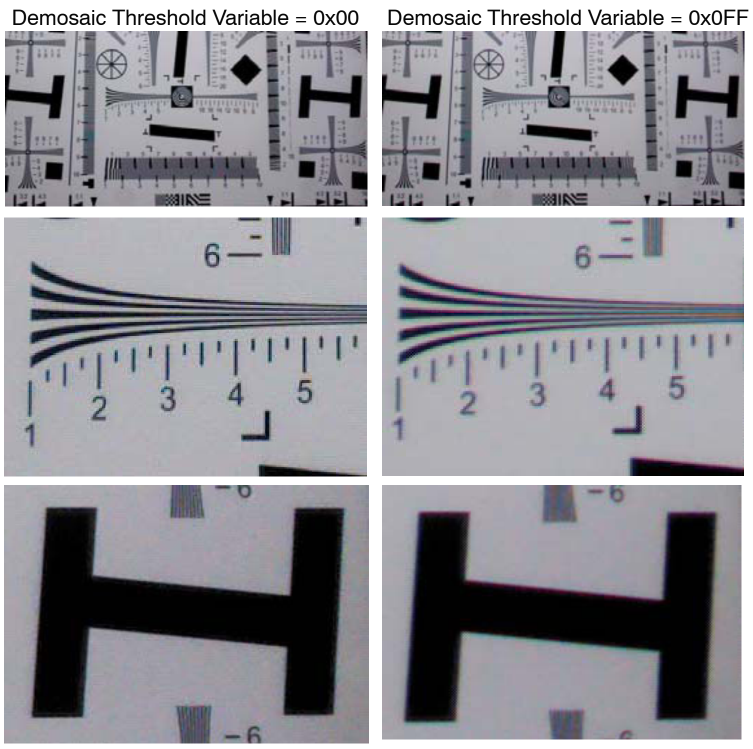

Demosaic Threshold Variable = 0x00
Demosaic Threshold Variable = 0x0FF

Figure 55. Demosaic Threshold Variable Min and max

4. Sharpness (cam_ll_start_ap_gain, cam_ll_stop_ap_gain)
   As the value of this variable increases, the amount of sharpness in the image will increase. It is expected that cam_ll_stop_ap_gain will be set to a lower value than cam_ll_start_ap_gain as when in low light the user will not want to apply a lot of sharpening as this would be sharpening noise and would not produce pleasing images.

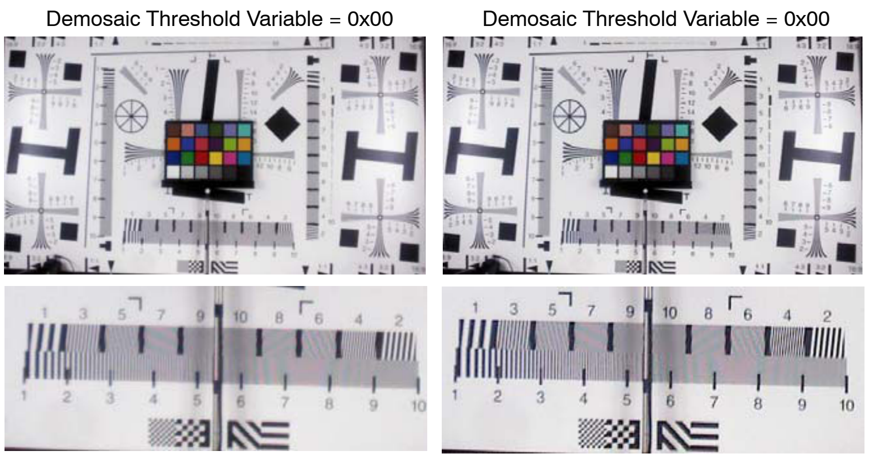

Demosaic Threshold Variable = 0x00
Demosaic Threshold Variable = 0x00

Figure 56. Sharpness Variable Min and Max

5. Sharpness Threshold (cam_ll_start_ap_thresh,cam_ll_stop_ap_thresh) This is the threshold at which the algorithm will consider a pixel for sharpening. As the value of this variable is increased, the image will blur as fewer pixels will be used for sharpening. It is expected that cam_ll_stop_ap_thresh is set greater than cam_ll_start_ap_thresh.
NOTE: If cam_ll_stop_ap_gain or cam_ll_start_ap_gain = 0x0, changing the threshold will have no effect.

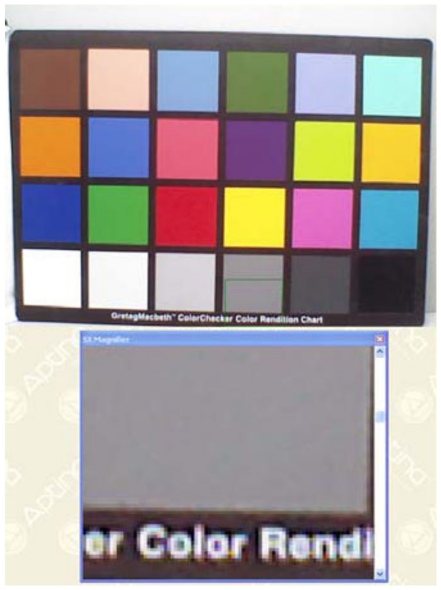


Figure 57. Sharpness Threshold – (0x00) Min

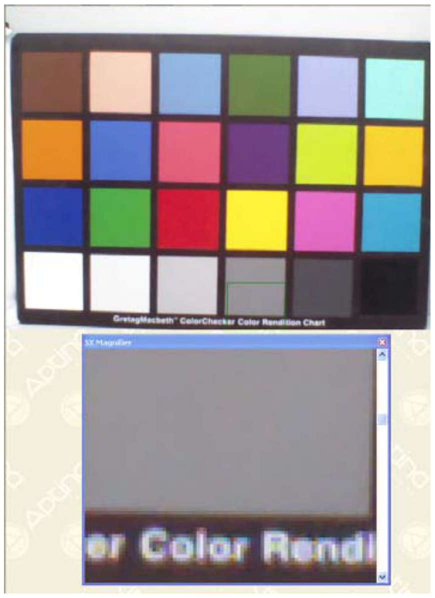


Figure 58. Sharpness Threshold – (0x0FF) Max

Cluster Defect Correction

The cluster defect correction algorithm is used in low light conditions to correct for defects. This correction algorithm is automatically enabled or disabled, depending on brightness metric and a threshold (cam_ll_cluster_dc_th_bm). This threshold has a hysteresis (*cam_ll_cluster_dc_gate_percentage*) to avoid possible oscillations that may happen if for instance the brightness metric is not stable due to the noise in the scene (quite normal in dark conditions). Figure 59 illustrates how the enable and disable of the cluster defect correction functions. Note that when enabling the cluster defect correction, the 2D aperture correction is automatically disabled.

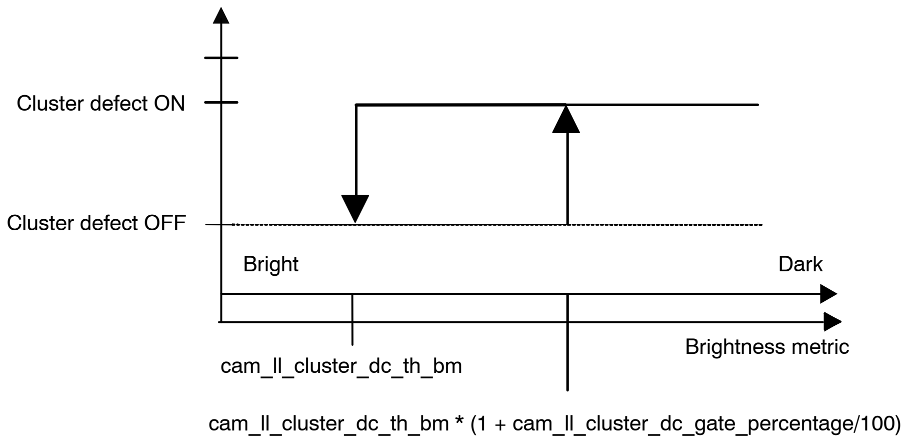


Figure 59. Cluster Defect Correction Functionality

Tuning of cam_ll_cluster_dc_th_bm

To set up the *cam_ll_cluster_dc_th_bm* point:

1. Set the camera up for desired frame rate and with 10 lux light (this light level is a guideline.)
2. Note the value of *cam_ll_inv_brightness_metric* and store this value into *cam_ll_cluster_dc_th_bm*
3. The default value of *cam_ll_cluster_dc_gate_percentage* should be acceptable, but the user may wish to fine−tune it further.

Figure 60 shows the effect cluster defect correction.

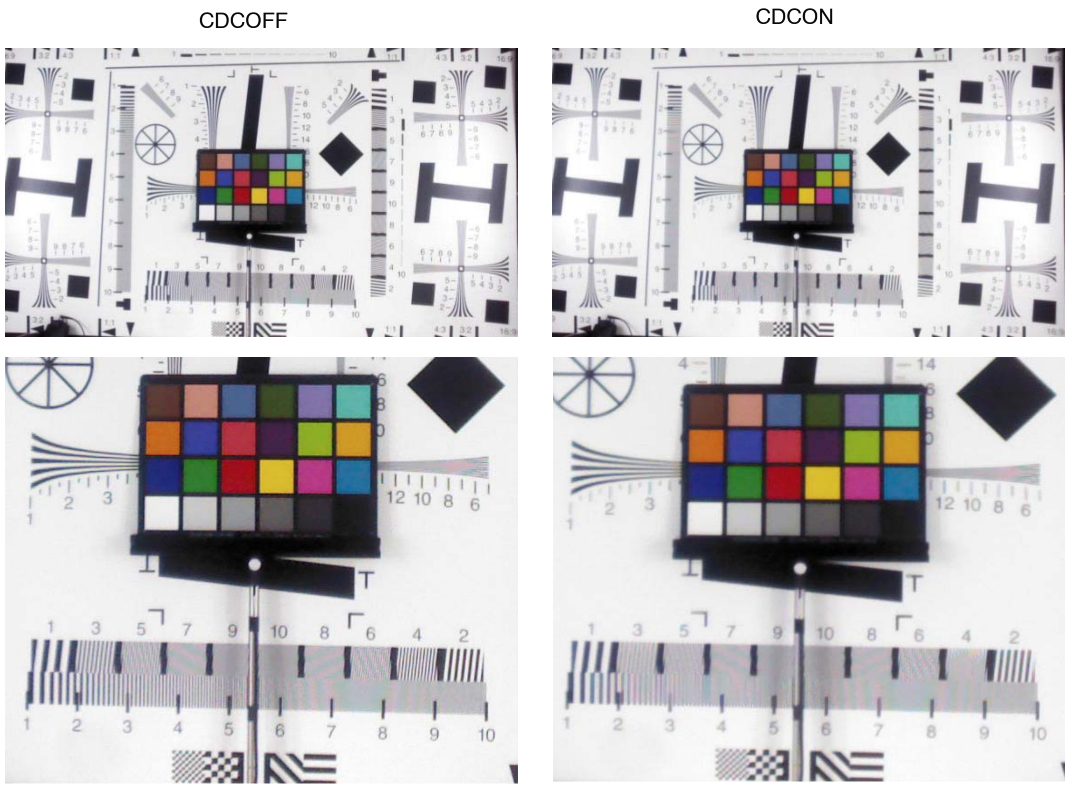


Figure 60. Effect Cluster Defect Correction

OVERLAY

Image Overlay

The ASX340AT supports an image overlay feature. Users are allowed to use up to five buffers for loading images, and display up to four layers for image overlay. Each layer can be configured with different properties including blinking rate, transparency, etc.

NOTE: Overlay is only available for NTSC/PAL output. It is not available for progressive output.

Image Format

Technically speaking, only run length encoding (RLE) format is supported for image overlay. However, DevWare supports different image formats for overlay including RLE, INI, BMP, and PNG because DevWare internally converts the images into the required RLE format. For overlay images stored in a flash or EEPROM device, it has to be in RLE format due to its compact size. DevWare FlashTool also only supports the RLE format for overlay images.

"Overlay Image Format Conversion" describes the details of how to convert images into the desired format using DevWare tools.

Image Overlay in DevWare

Image overlay GUI is under *Control →Graphics Overlay* as shown in Figure 61.

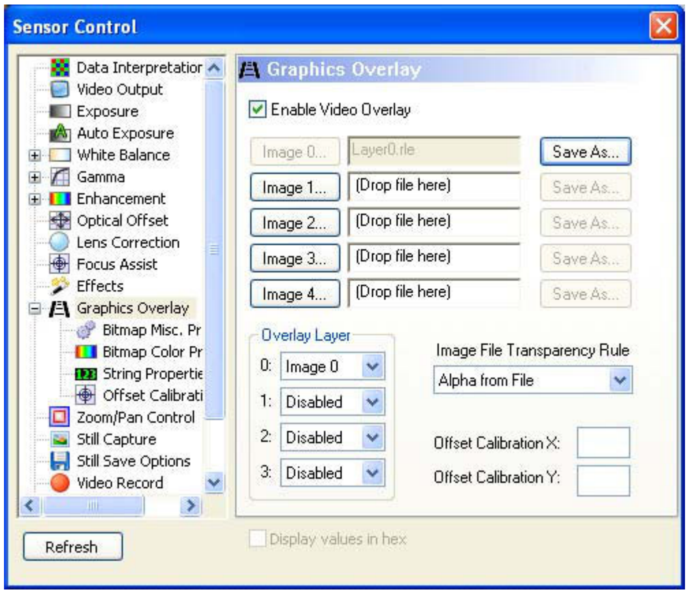


Figure 61. Image Overlay GUI in DeWARe

For example, to enable an image in the Layer 0 for overlay, do the following:

1. Go to *Control →Graphics Overlay*
2. Check *Enable Video Overlay*
3. Load an image for *Image 0* (this is actually loaded to *Buffer 0).*
4. In *Overlay Layer,* choose *Image 0* for *0:*.

Once a buffer is enabled by an overlay layer, users are able to save the overlay images into different formats by clicking *Save As* as shown in Figure 61. This is one way of converting image formats provided by DevWare. More details are discussed in "Overlay Image Format Conversion".

Properties of each image are configurable through *Control →Graphics Overlay →Bitmap Misc. Properties* as shown in Figure 62.

The overlay's position on the image is adjustable through *Horizontal Position* and *Vertical Position*. Due to the RLE format compression, the full overlay horizontally must be within the image at all times. Otherwise, the overlay will be shown as corrupted. However, overlays can be positioned vertically on any row within the image.

Blink Rate is in the unit of multiples of 5 frames. Timeout is also in the unit of multiples of 5 frames, and will disable the image overlay after the timeout period.

For Cropping, the position with (0,0) offset is the upper left corner of the image. The rectangular area defined by Cropping parameters is either kept when choosing *Crop Outside*, or removed when choosing *Crop Inside*.

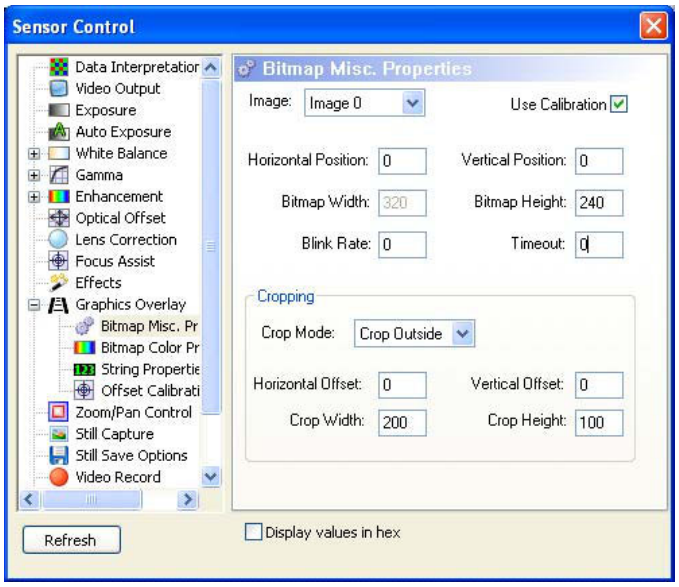


Image: Image 0
Use Calibration ✓
Horizontal Position: 0
Vertical Position: 0
Bitmap Width: 320
Bitmap Height: 240
Blink Rate: 0
Timeout: 0
Crop Mode: Crop Outside
Horizontal Offset: 0
Vertical Offset: 0
Crop Width: 200
Crop Height: 100

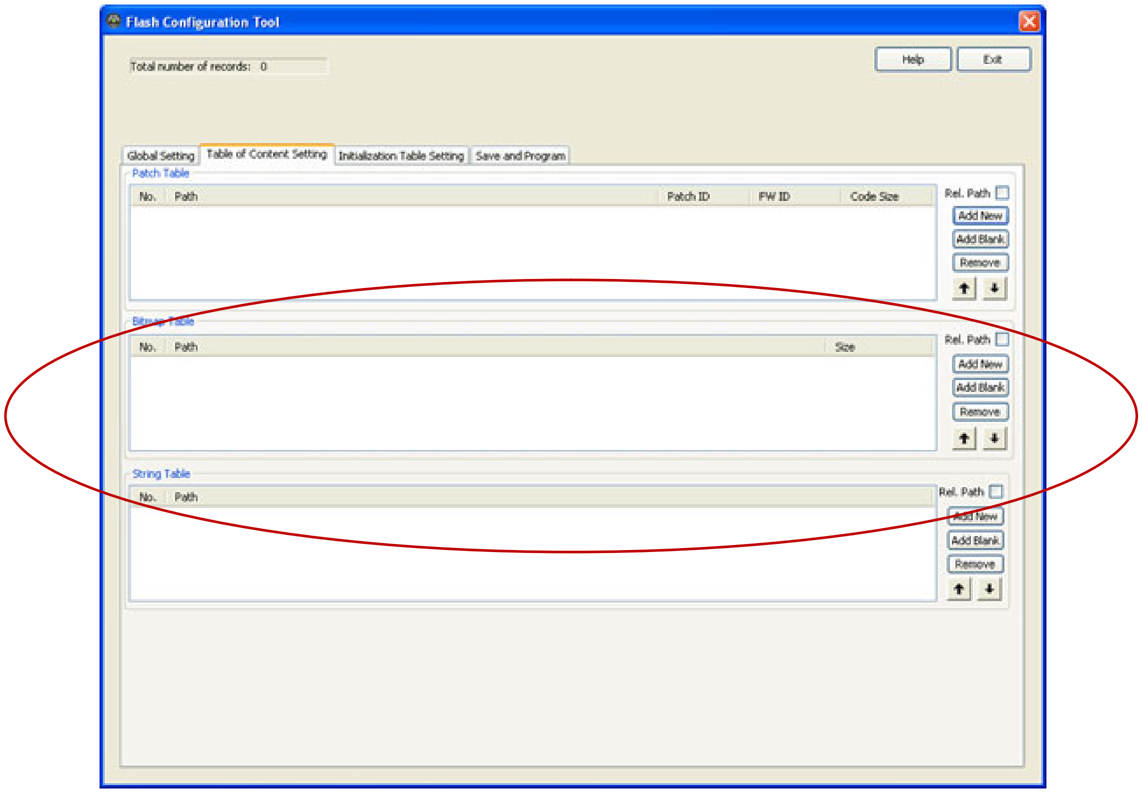
Figure 62. Overlay Image Properties Configuration GUI

Image Overlay in Flash/EEPROM

Images to be included in Flash/EEPROM have to be in RLE format. FlashTool by DevWare only supports RLE format for overlay images.

In FlashTool, overlay images are loaded to the *Bitmap Table* on *Table of Content Setting* tab as shown in Figure 63.

There is virtually no limit as to how many images can be loaded into the Bitmap Table as long as they don't overflow the flash.



Total number of records: 0

Figure 63. Bitmap Table Overlay Images in FlashTool

Table 36 specifies the flash commands for overlay including image overlay and string overlay. Refer to the ASX340AT Host Command Interface document for more details on each command.

Table 36. VARIABLES USED FOR FADE−TO−BLACK

Overlay Host Command	Value	Type	Description
Enable Overlay	0x8200	Synchronous	Enable or disable the overlay subsystem
Get Overlay State	0x8201	Synchronous	Retrieves the state of the overlay subsystem
Set Calibration	0x8202	Synchronous	Set the calibration offset
Set Bitmap Property	0x8203	Synchronous	Set a property of a bitmap
Get Bitmap Property	0x8204	Synchronous	Get a property of a bitmap
Set String Property	0x8205	Synchronous	Set a property of a character string
Load Buffer	0x8206	Asynchronous	Load an overlay buffer with a bitmap (from EPROM/Flash)
Load Status	0x8207	Synchronous	Retrieve status of an active load buffer operation
Write Buffer	0x8208	Synchronous	Write directly to an overlay buffer
Read Buffer	0x8209	Synchronous	Read directly from an overlay buffer
Enable Layer	0x820A	Synchronous	Enable or disable an overlay layer
Get Layer Status	0x820B	Synchronous	Retrieve the status of an overlay layer
Set String	0x820C	Synchronous	Set the character string
Get String	0x820D	Synchronous	Get the current character string.
Load String	0x820E	Asynchronous	Load a character string (from Flash)

Note that one case for overlay in Flash/EEPROM should be avoided: Load_Buffer with display set to True, followed by an Enable_Layer command with display set to True.

Therefore, for successful image overlay, there are two approaches:

- #1: Load_Buffer with display set to True.
  This will display the overlay image. This is a simple and straightforward approach, and is useful if no image properties need to be set before displaying the image.
- #2: Load_Buffer first with display set to False, then use Enable_Layer with display set to True for displaying.
  This approach is desired when there are bitmap properties that need to be set before displaying the image. Put the set_bitmap_property command before the Enable_Layer command.

Flash command examples are provided below for #1 and #2 approaches.

- #1 approach:

```
[OVERLAY_INIT_TABLE]
TYPE=create_command
PARAMETERS={ Enable_Overlay, 0x1 }
TYPE=create_command
PARAMETERS={ Load_Buffer, 0x0, 0x0, 0x0,
0x1 }
TYPE=create_command
PARAMETERS={ Load_Buffer, 0x1, 0x1, 0x1,
0x1 }
[END] # [OVERLAY_INIT_TABLE]
```

- #2 approach:

```
[OVERLAY_INIT_TABLE]
TYPE=create_command
PARAMETERS={ Enable_Overlay, 0x1}
TYPE=create_command
PARAMETERS={ Load_Buffer, 0x0, 0x0, 0x0,
0x0 }
TYPE=create_command
PARAMETERS={ Enable_Layer, 0x0, 0x0, 0x1 }
TYPE=create_command
PARAMETERS={ Load_Buffer, 0x1, 0x1, 0x1,
0x0 }
TYPE=create_command
PARAMETERS={ Enable_Layer, 0x1, 0x1, 0x1 }
[END] # [OVERLAY_INIT_TABLE]
```

Buffer 4 Usage Limitation

ASX340AT has five buffers − Buffer 0 to Buffer 4. All buffers can be used for loading overlay images. However, there is a limitation on Buffer 4.

Buffer 4 is reserved by firmware during System Initialization. This means, users cannot use Buffer 4 during the System Initialization phase, i.e. before system streaming or start of frame (SOF). This means, when configuring a fcfg file, Buffer 4 cannot be used with a Load_Buffer host command within Global, Patch, Calibration, or Graphics Overlay sections as shown in Figure 69: "Initialization Table Setting Tab". Buffer 4 can be used, however, in a command sequence, which is only invoked after SOF. Such a command sequence can be triggered with GPIO pin (Set_GPI_Association), or SOF event (Event_Set_Association); however, it cannot be triggered with Invoke_Command_Seq command since a command sequence will be executed during System Initialization phase if it is triggered with Invoke_Command_Seq command.

Overlay Image Format Conversion

For a single image file conversion, go to *Control* → *Graphics Overlay.* Once an image is loaded and also enabled by a layer, as shown in Figure 61, the *Save as* button is enabled and can be used to save the current image into other supported formats.

DevWare also provides a command line tool − Makeover − for converting images between different formats. It supports RLE, INI, BMP, and PNG image formats.

Refer to *Start* → **onsemi** *Imaging* → *Tools* → *Command Line Tools* → *Makeover User Guide* for detailed information on this tool. The Makeover command line tool is accessible in DOS. In a DOS command window, type *makeover* and hit *Enter*, it will display the usage of this command as well.

An example of converting a RLE into equivalent INI commands is shown as follows:

- Open DOS command window (go to *Start* → *Run*, and type *cmd*, and hit *Enter*)
- Change the directory into where the RLE image is located (use *cd* dos command to change directory)
  To convert *sample.rle* to *sample.ini*, use the following command:

```
makeover -inrle sample.rle -outini sample.ini
```

This will convert s*ample.rle* into *sample.ini* saved in the current directory.

To do a batch process of converting a number of files at one time, a for loop can be used under DOS:

- Convert all RLE files to BMP with the same base names:

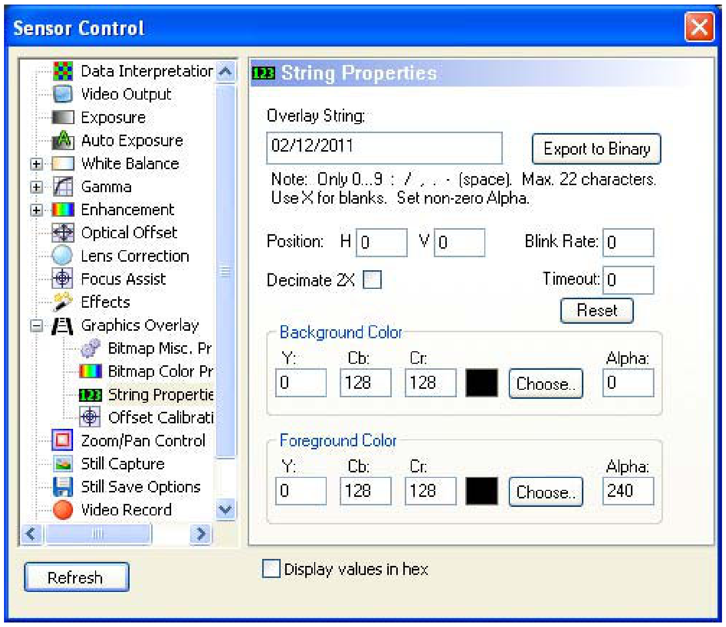
```
for %I in (*.rle) do makeover -inrle %I -outbmp
%~nI.bmp
```

- Convert all BMP files to RLE with the same base names:

```
for %I in (*.bmp) do makeover -inbmp %I -outrle
%~nI.rle
```

Overlay Strings in Binary Format

The ASX340AT also supports a string overlay feature. In DevWare, go to *Control* → *Graphics Overlay* → *String Properties*. As shown in Figure 64, the overlay string can be input in the box. String properties are also configurable on this GUI, including blink rate, color, position, etc. Please note that overlay needs to be enabled (see Figure 61) for a string to be displayed on the video.



Overlay String: 02/12/2011

Note: Only 0...9 : / , . - (space). Max. 22 characters.
Use X for blanks. Set non-zero Alpha.

Position: H 0 V 0 Blink Rate: 0
Decimate 2X ☐ Timeout: 0

Background Color: Y: 0 Cb: 128 Cr: 128 Alpha: 0
Foreground Color: Y: 0 Cb: 128 Cr: 128 Alpha: 240

☐ Display values in hex

Figure 64. String Overlay in DevWare

Export to Binary option in Figure 64 saves the input string into a binary file. Strings are required to be in binary format when they are included in Flash/EEPROM. In FlashTool, overlay strings are loaded to the String Table on the Table of Content Setting tab as shown in Figure 63. There is virtually no limit as how many strings can be loaded into the String Table as long as they don't overflow the flash.

String overlay commands are also summarized in Table 36. Please refer to the ASX340AT Host Command Interface document for more details on each command.

HOST COMMAND INTERFACE

Figure 65 shows the hardware context of the host command interface for the ASX340AT. The host controls the ASX340AT through its two−wire interface bus. This gives the host access both to the internal RAM of the device, and to its peripherals. However, the intention is that the host will primarily interact with the device through its Command Register.

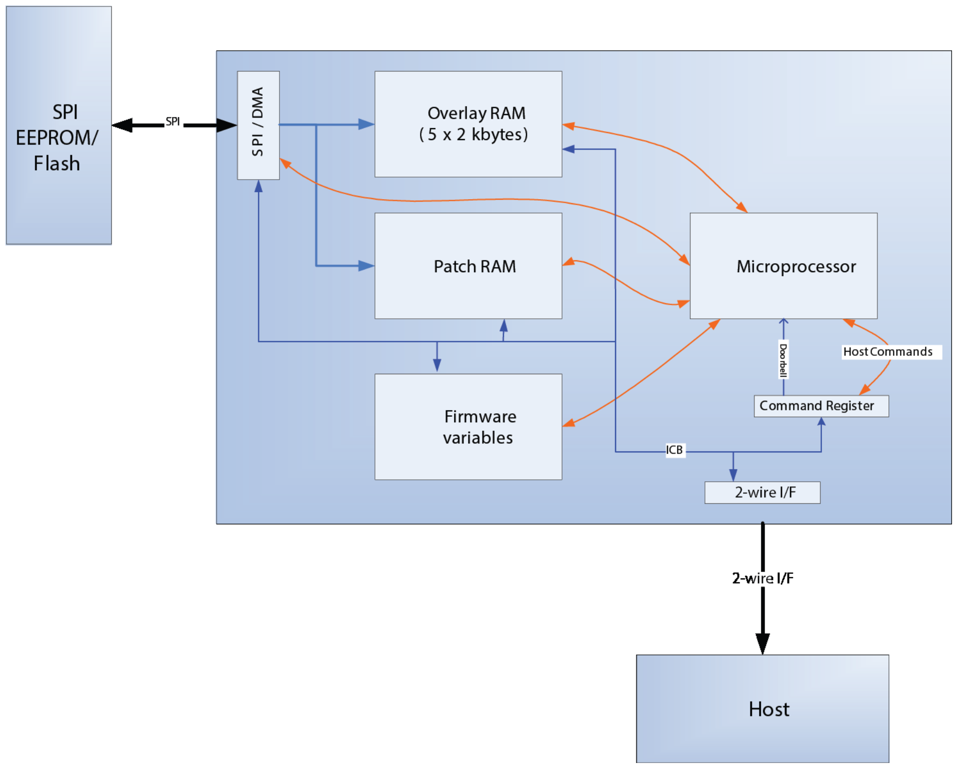


Figure 65. Host Command Interface Context Block Diagram

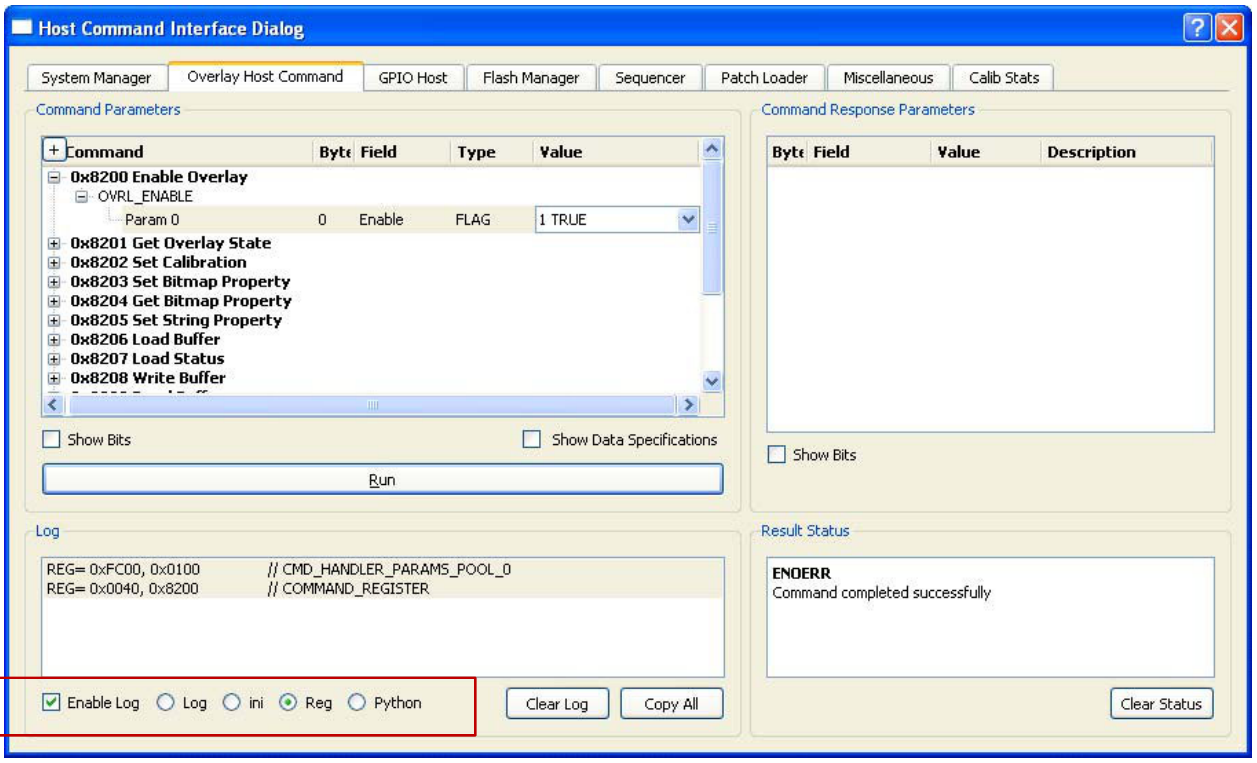
Please refer to the first few pages of the Host Command Interface document for more details on the host command interface architecture, processing flow, and all the available host commands.

Flash Supported Host Commands

A subset of the host commands is supported by FlashTool for ASX340AT. Appendix H of the Host Command Interface document specifies the host commands that are supported within command sequences. There are special restrictions/instructions on change−config commands as documented in Appendix H. Examples are given in“Change Config Command”.

DevWare HCI Plug−in

DevWare offers the Host Command Interface Dialog under Plug−ins as shown in Figure 66.



Command	Byte	Field	Type	Value
0x8200 Enable Overlay				
OVRL_ENABLE				
Param 0	0	Enable	FLAG	1 TRUE
0x8201 Get Overlay State				
0x8202 Set Calibration				
0x8203 Set Bitmap Property				
0x8204 Get Bitmap Property				
0x8205 Set String Property				
0x8206 Load Buffer				
0x8207 Load Status				
0x8208 Write Buffer				

Byte	Field	Value	Description

```
REG= 0xFC00, 0x0100          // CMD_HANDLER_PARAMS_POOL_0
REG= 0x0040, 0x8200          // COMMAND_REGISTER
```

```
ENOERR
Command completed successfully
```

Figure 66. Host Command Interface Plug−in

Host commands are categorized into different tabs based on their functionalities. To send a host command, choose the corresponding command, and fill in the parameter value(s), then click *Run*. The host commands can be captured in log, ini, register, or Python formats by checking *Enable Log* as highlighted in the red box.

HCI Command Example for Writing/Reading from Flash/EEPROM

Using the HCI plug−in, users are able to do some simple read/write to Flash/EEPROM devices. The NVM device related commands are available under the "Flash Manager" tab. All commands apply to both flash and EEPROM except the Erase Block and Erase Device, which are only applicable to Flash.

Below is an example of accessing an NVM device. These commands can be sent through two−wire interface as well. Two−wire interface commands are provided with FIELD_WR under each step.

1. Get Lock

FIELD_WR = COMMAND_REGISTER, HOST_COMMAND, 0x8500

2. Lock Status (check that lock was successfully obtained)

FIELD_WR = COMMAND_REGISTER, HOST_COMMAND, 0x8501

3. Configure Device

FIELD_WR = CMD_HANDLER_PARAMS_POOL_0, 0x0503 // ID=5, read cmd = 03

FIELD_WR = CMD_HANDLER_PARAMS_POOL_1, 0x0318 // width=3, bits=24

FIELD_WR = CMD_HANDLER_PARAMS_POOL_2, 0x0020 // size = 2Mbytes

IELD_WR = CMD_HANDLER_PARAMS_POOL_3, 0x0000

FIELD_WR = COMMAND_REGISTER, HOST_COMMAND, 0x850A

- logical ID = use 5 (or less, but 5 works for all flashes) for flash, more than 5 for EEPROM
- read cmd = 3
- address width = number of bytes in address (address bits divided by 8, rounded up)

- address bits = number of bits in address
- size = size of memory in bytes

4. Erase Device (wait for command to complete, only applies to flash)
   FIELD_WR = COMMAND_REGISTER, HOST_COMMAND, 0x8507
5. Flash Status (check erase completed)
   FIELD_WR = COMMAND_REGISTER, HOST_COMMAND, 0x8509
6. Read (read 16 bytes from address 0)
   FIELD_WR = CMD_HANDLER_PARAMS_POOL_0, 0x0000
   FIELD_WR = CMD_HANDLER_PARAMS_POOL_2, 0x1000
   FIELD_WR = COMMAND_REGISTER, HOST_COMMAND, 0x8504
7. Get Status
   FIELD_WR = COMMAND_REGISTER, HOST_COMMAND, 0x8509
   - Returns 0xFF in all 16 response bytes
8. Write (write the first 4 bytes at address 0. Note in the HCI plug−in, specify 0 for all the unused parameters)
   FIELD_WR = CMD_HANDLER_PARAMS_POOL_0, 0x0000
   FIELD_WR = CMD_HANDLER_PARAMS_POOL_2, 0x04AA
   FIELD_WR = CMD_HANDLER_PARAMS_POOL_3, 0xBBCC
   FIELD_WR = CMD_HANDLER_PARAMS_POOL_4, 0xDD00
   FIELD_WR = CMD_HANDLER_PARAMS_POOL_5, 0x0000
   FIELD_WR = CMD_HANDLER_PARAMS_POOL_6, 0x0000
   FIELD_WR = CMD_HANDLER_PARAMS_POOL_7, 0x0000
   FIELD_WR = COMMAND_REGISTER, HOST_COMMAND, 0x8505
9. Get Status (check write completed)
   FIELD_WR = COMMAND_REGISTER, HOST_COMMAND, 0x8509
10. Read (read 16 bytes from address 0)
   FIELD_WR = CMD_HANDLER_PARAMS_POOL_0, 0x0000
   FIELD_WR = CMD_HANDLER_PARAMS_POOL_2, 0x1000
   FIELD_WR = COMMAND_REGISTER, HOST_COMMAND, 0x8504
11. Get status
   FIELD_WR = COMMAND_REGISTER, HOST_COMMAND, 0x8509
   - Returns:
     [0] 0xAA
     [1] 0xBB
     [2] 0xCC
     [3] 0xDD
     [4..15] 0xFF
12. Release Lock
   FIELD_WR = COMMAND_REGISTER, HOST_COMMAND, 0x8502

FLASH/EEPROM PROGRAMMING

The ASX340AT supports SPI nonvolatile memory (NVM) devices including flash and EEPROM for storing records/commands. DevWare provides FlashTool (C:\onsemi Imaging\Tools\FlashTool) for programming the memory devices.

Supported NVM Devices

The ASX340AT supports a variety of SPI NVM devices. Table 37 gives some examples.

Table 37. EXAMPLES OF SUPPORTED SPI NVM DEVICES

Logical ID	Manufacturer	Device	Type	Size	Autodetected by FirmWare	ManualD DeviceD1 DeviceD2
1	Atmel	AT26DF081A	Flash	1 Mbyte	Yes	0x1F4501
3	Sanyo	LE25FW806	Flash	1 Mbyte	Yes	0x622662
4	ST	M25P05A	Flash	64 kbyte	Yes	0x202010
5	ST	M25P16	Flash	2 Mbyte	Yes	0x202015
6	ST	M95040	EEPROM	512 byte	No	0x20FFFF
7	ST	M95020	EEPROM	256 byte	No	0x20FFFF
8	ST	M95010	EEPROM	128 byte	No	0x20FFFF
9	ST	M95M01	EEPROM	128 byte	No	0x20FFFF
10	ST	M25AA080	EEPROM	1 kbyte	No	0x29FFFF
11	Microchip	M25LC080	EEPROM	1 kbyte	No	0x29FFFF

For SPI Flash, the ASX340AT supports any flash device that satisfies the following criteria:

- The flash device uses 24−bit address;
- The flash device supports the following commands:

Table 38. FLASH COMMAND LIST

Command	Opcode
Write Status	0x01
Read	0x03
Read Status	0x05
Write Enable	0x06
Fast Read	0x0B
Page Program	0x02
Block Erase	0xD8
Chip Erase	0xC7
Read Manufacturer and Device ID	0x9F

For EEPROM, the ASX340AT currently supports any device that has a 7, 8, 9, 16, or 24−bit address width.

NVM Device Discovery and Configuration

The ASX340AT uses the 0x9F *Query Device* command to determine if a JEDEC−compliant Flash device is attached. If this command is ignored (i.e. the attached device doesn't return any data) because it is not JEDEC−compliant, then 8 bytes will be read from address 0x0, assuming a 24−bit address.

The ASX340AT will then search for the TOC *version string* entry within the returned data. Where the data appears (if at all) determines whether the device has a 7−, 8−, 9−, 16−, or 24−bit address width.

The procedure is outlined below:

1. Issue 0x9F *Query Device* command and read 3 bytes
2. Determine if device is supported,
    - If so, go to Flash−Config
    - Otherwise,
      a. Read 8 bytes from address 0x0
      (24−bit address, i.e. 3 bytes transmitted)
      b. Check returned data for TOC 'version string'
        If 'version string' is present, go to Flash−Config
        If returned data is all 0x0s, go to Host−Config
        If returned data is not all 0x0s, go to Auto−Config

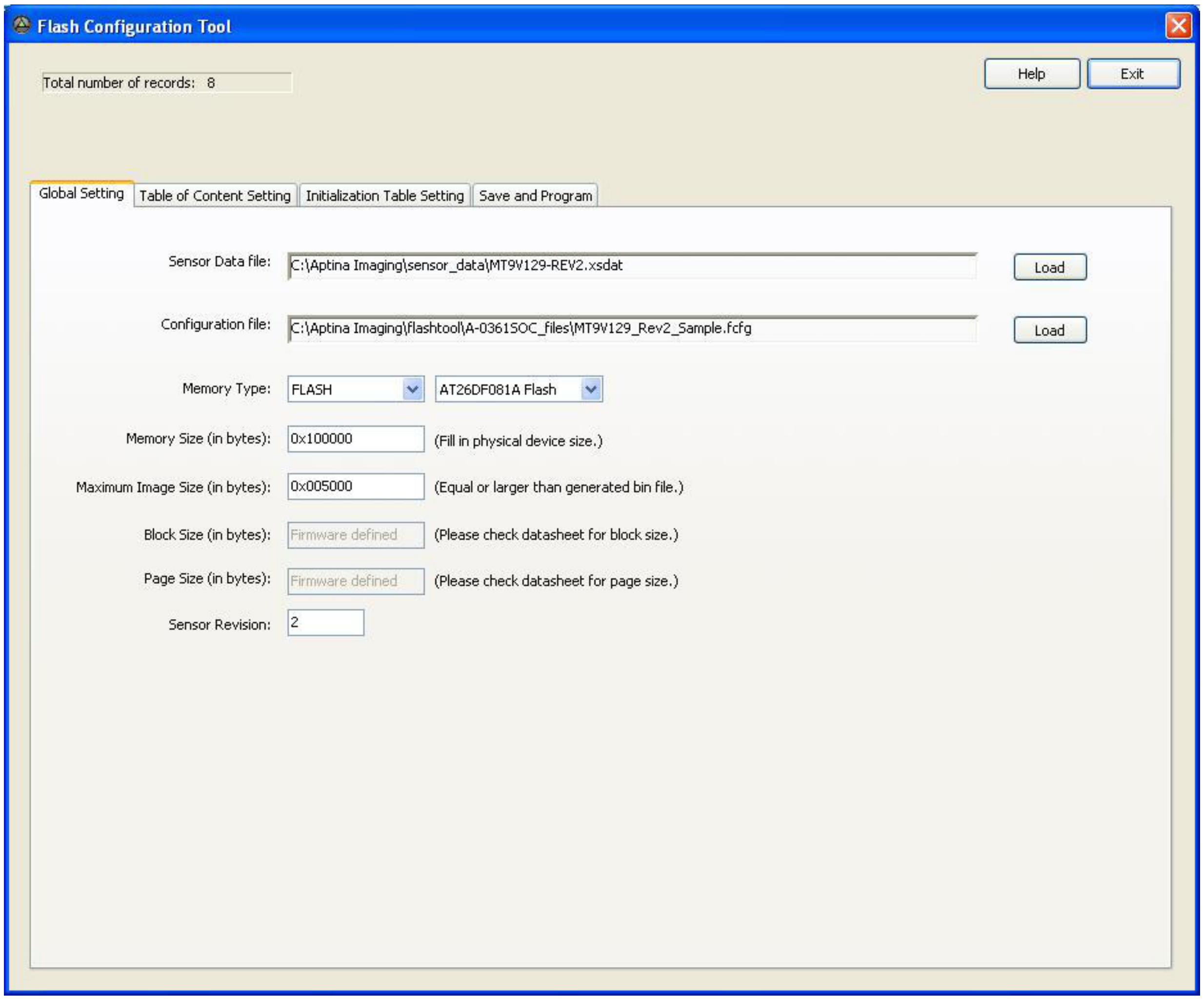
If users wish to emulate an SPI NVM device, it depends if they want to program it via the ASX340AT command interface for the best emulation mechanism:

- If users want to erase and program it via commands: emulate a JEDEC−compliant device, for example, M25P16. The emulation will respond to the 0x9F command with the same data as the M25P16.

- If users just want to read it: emulate a 24−bit addressed EEPROM. This will not respond to the 0x9F “query” device command.

To properly program an NVM device in Flash tool, users need to specify the correct device type. Figure 67 shows the Global Setting tab, where the user specifies the configuration of the NVM device according to the following:

- Memory Type
  The user specifies whether the NVM device is a flash or EEPROM.
- Memory Size
  This is the physical size of the NVM device. Users may need to enter this manually if the specific NVM device is not in a drop−down list from Memory Type selection.
- Maximum Image Size
  This is the maximum size allowed on the NVM device to be programmed with the flash bin file. Therefore, it needs to be smaller than or equal to Memory Size, but larger than or equal to the bin file generated or used for programming. This needs to be entered by the user.
- Block Size
  This only applies to flash devices, and it is typically specified in a flash device’s data sheet. If multiple blocks are defined for a flash device, it is the one that corresponds to the block erase command 0xD8. It needs to be manually entered by the user if “standard flash” option is chosen.
- Page Size
  This only applies to flash devices, and it is typically specified in a flash device’s data sheet. It needs to be manually entered by the user if “standard flash” option is chosen.



Total number of records: 8

Sensor Data file: C:\Aptina Imaging\sensor_data\MT9V129-REV2.xsdat

Configuration file: C:\Aptina Imaging\flashtool\A-0361SOC_files\MT9V129_Rev2_Sample.fcfg

Memory Type: FLASH AT26DF081A Flash

Memory Size (in bytes): 0x100000 (Fill in physical device size.)

Maximum Image Size (in bytes): 0x005000 (Equal or larger than generated bin file.)

Block Size (in bytes): Firmware defined (Please check datasheet for block size.)

Page Size (in bytes): Firmware defined (Please check datasheet for page size.)

Sensor Revision: 2

Figure 67. Flash Tool − Global Setting

If a flash device is present, refer to Figure 67. The user selects “Flash” in the first drop−down list and the specific flash name from the second drop−down list.

If the flash device is not in the second drop−down list, select “standard flash” option instead. The user then needs to enter the Memory Size, Maximum Image Size, Block Size and Page Size manually on the GUI. For standard flash devices, the maximum size that can be programmed is 2M byte. That is, even if a flash device is larger than 2M byte in size, the flash tool can only program up to 2M byte of that flash device. Logical ID 5 should be used for programming standard flash devices.

If an EEPROM device is present, refer to Figure 67. For Memory Type, the user selects “EEPROM” in the first drop−down list and the corresponding number of address bits from the second drop−down list (this is typically specified in the device’s data sheet). Memory Size and Maximum Image Size are both required for user input as well.

Head Board Flash/EEPROM Configuration

onsemi builds the ASX340AT head board with the following two memory devices to be used in Flash mode:

- U14 = M95M01 (EEPROM)
- U16 = M25P05 (Flash)

P31 controls which memory device is connected/chosen for programming or booting up:

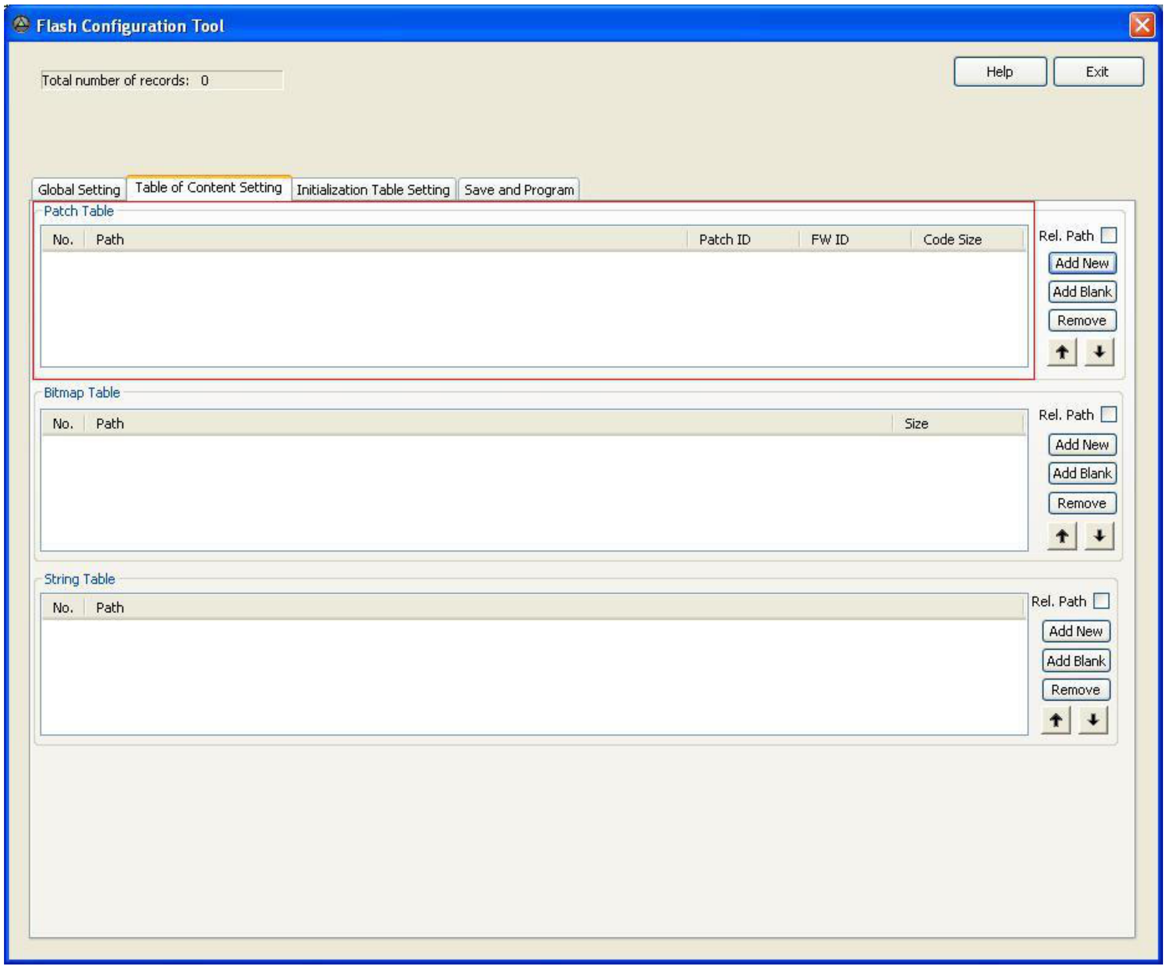
- If the jumper connects 1 and 2 on P31, U14 is selected;
- If the jumper connects 2 and 3 on P31, U16 is selected.

The jumper on P28 needs to connect 1 and 2 for the head board to be in flash configuration mode.

Patches

In the case when patches are required to apply, they need to be included in Patch Table as shown in Figure 68. However, including a patch in the Patch Table does not mean it is automatically applied. In order to apply a patch, the *Load_Patch <Patch ID>* command is required to be included in *Initialization Table Setting* → *Patch* section as shown in Figure 69. The patch ID corresponds to the number in Patch Table.

For information on applicable patches, refer to the Firmware Release Notes document.



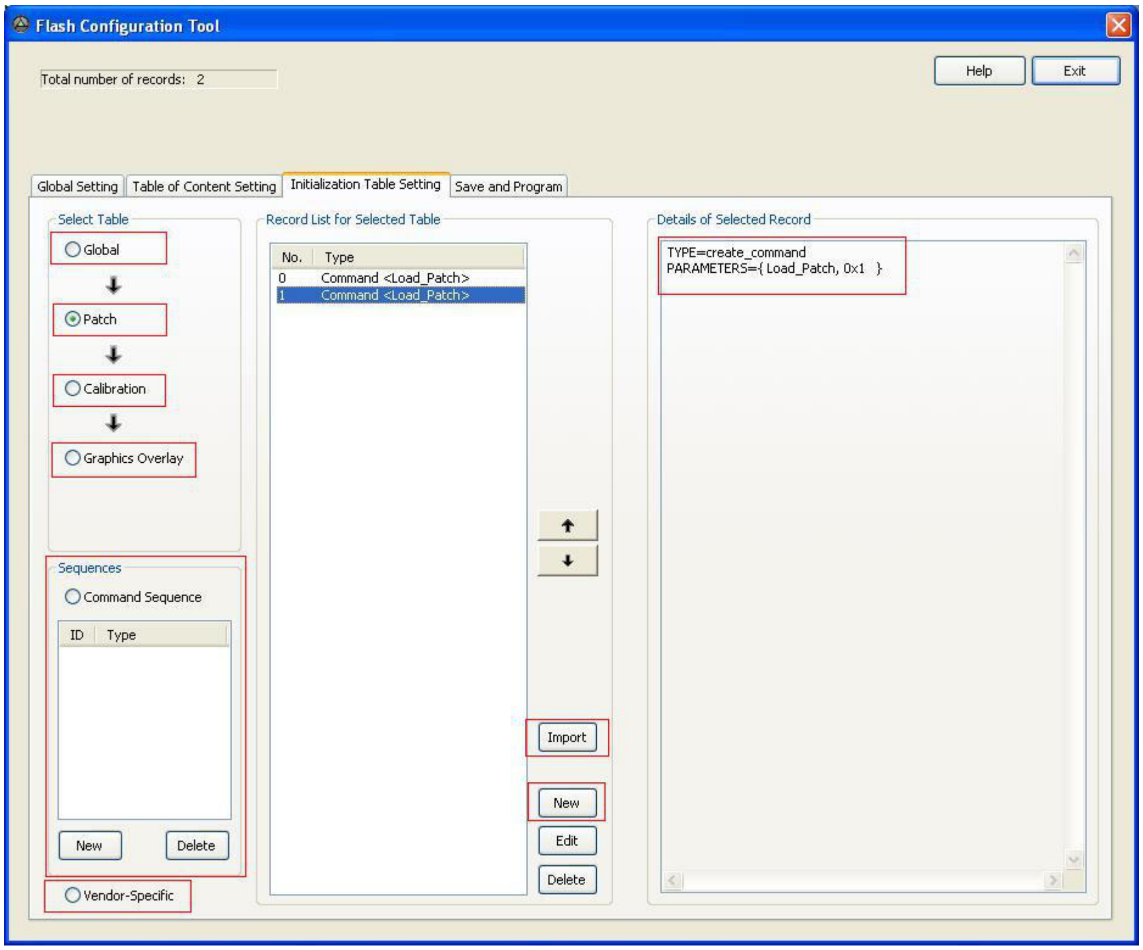
Total number of records: 0

No.	Path	Patch ID	FW ID	Code Size

No.	Path	Size

No.	Path

Figure 68. Patch Table



Total number of records: 2

No.	Type
0	Command <Load_Patch>
1	Command <Load_Patch>

```
TYPE=create_command
PARAMETERS={ Load_Patch, 0x1  }
```

Figure 69. Initialization Table Setting Tab

Image and String Overlay

Refer to “Image Overlay in Flash/EEPROM” for configuring image and string overlay in FlashTool.

Convert INI Commands to Flash Records

All the register writes and commands are specified in their proper sections on the **Initialization Table Setting** tab as shown in Figure 69. To add one command line at a time, click on New, which is mostly useful for adding an HCI command as shown in Figure 70. There are other record types that can be selected in the drop−down list of *Select Record Type*, such as customer−specific records, and so on.

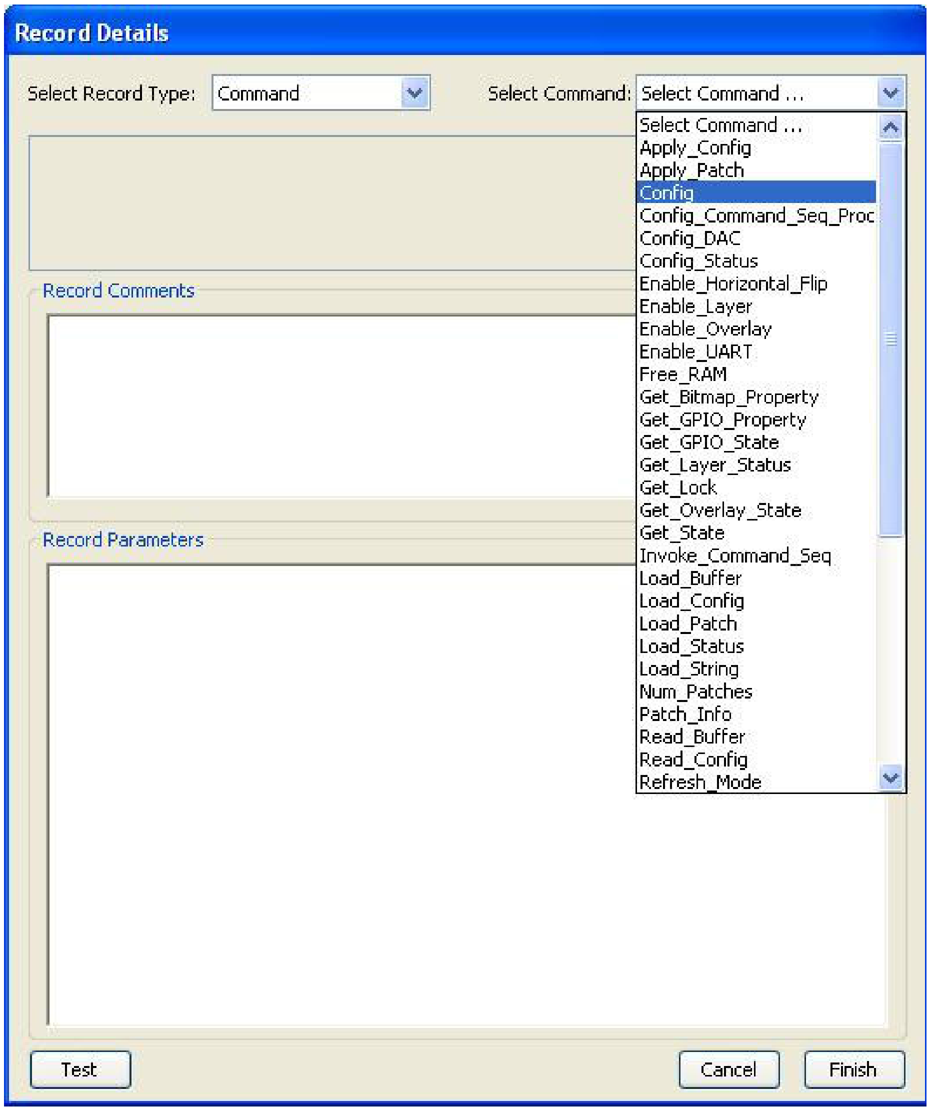


Figure 70. Add New Command Line in FlashTool

For direct register writes or a set of register writes such as CCM, AWB, and etc., users can capture them first by using the logging feature in DevWare, and then converting them into Flash records. (Note that direct writes to the Command Registers should not be included in flash records. Refer to Host Command Interface document for more details.) Two examples on the procedure are provided below.

To convert commands for changing image orientation to Horizontal Mirror:

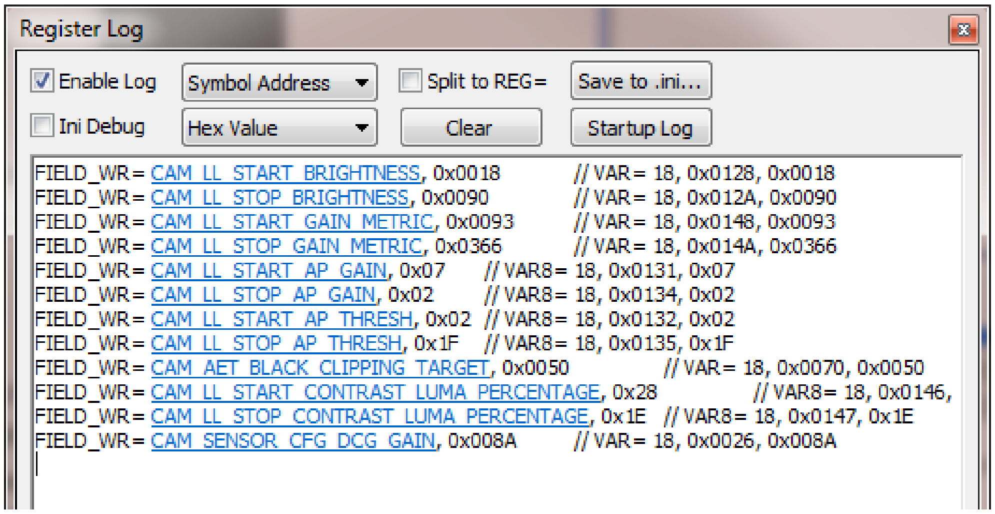
1. In DevWare, open the *Log* window and check only *Enable Log* as shown in Figure 71.
2. Go to *Control* → *Video Output*, and change the orientation to *Horizontal Mirror*, and click *Apply*.
3. Copy all the lines in the Log window.
4. Go to FlashTool, and click on Import as shown in Figure 69 to bring up Figure 72 GUI.
5. Click on *Load Text* and paste command lines.
6. Click on *Convert* and they are ready to be added to flash records.

When users apply new settings by loading an ini file in DevWare, the same procedure applies to capture a set of register writes to convert to flash records.

To convert AWB and CCM settings captured from Sensor Tune tool to flash records:

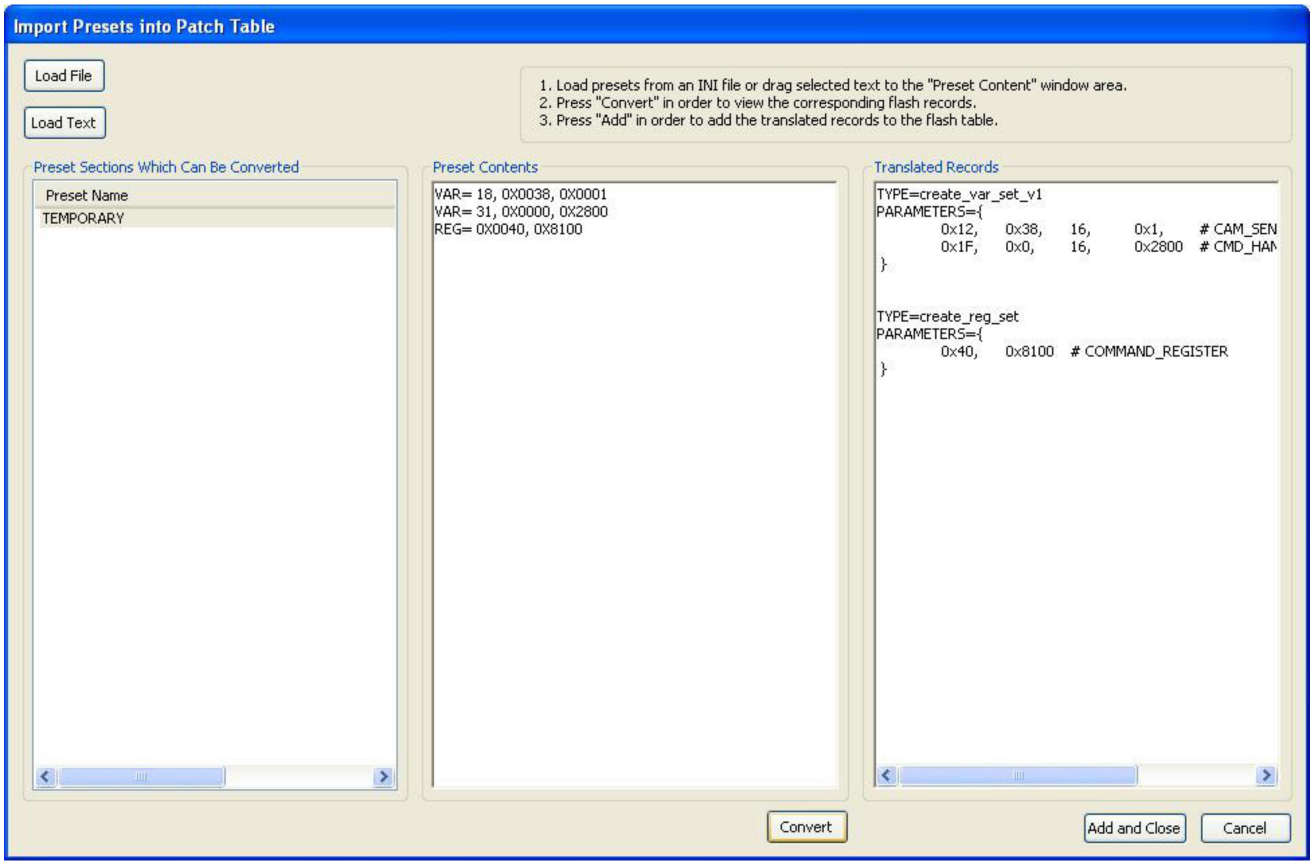
1. Save the AWB and CCM settings generated by Sensor Tune tool to an ini file.
2. Open the AWB and CCM ini file in DevWare, highlight the preset, and click on *Reg* → *Var* to generate a new preset labeled with REG →VAR.
3. Open the Log window and check only *Enable Log* as shown in Figure 71.
4. Execute the new AWB and CCM preset labeled with REG →VAR.
5. Copy all the lines in the Log window.
6. Go to FlashTool, and click on *Import* as shown in Figure 69 to bring up Figure 72 GUI.
7. Click on *Load Text* and paste command lines.

8. Click on *Convert* and they are ready to be added to flash records.



```
FIELD_WR= CAM_LL_START_BRIGHTNESS, 0x0018        // VAR= 18, 0x0128, 0x0018
FIELD_WR= CAM_LL_STOP_BRIGHTNESS, 0x0090         // VAR= 18, 0x012A, 0x0090
FIELD_WR= CAM_LL_START_GAIN_METRIC, 0x0093       // VAR= 18, 0x0148, 0x0093
FIELD_WR= CAM_LL_STOP_GAIN_METRIC, 0x0366        // VAR= 18, 0x014A, 0x0366
FIELD_WR= CAM_LL_START_AP_GAIN, 0x07      // VAR8= 18, 0x0131, 0x07
FIELD_WR= CAM_LL_STOP_AP_GAIN, 0x02       // VAR8= 18, 0x0134, 0x02
FIELD_WR= CAM_LL_START_AP_THRESH, 0x02    // VAR8= 18, 0x0132, 0x02
FIELD_WR= CAM_LL_STOP_AP_THRESH, 0x1F     // VAR8= 18, 0x0135, 0x1F
FIELD_WR= CAM_AET_BLACK_CLIPPING_TARGET, 0x0050          // VAR= 18, 0x0070, 0x0050
FIELD_WR= CAM_LL_START_CONTRAST_LUMA_PERCENTAGE, 0x28          // VAR8= 18, 0x0146,
FIELD_WR= CAM_LL_STOP_CONTRAST_LUMA_PERCENTAGE, 0x1E   // VAR8= 18, 0x0147, 0x1E
FIELD_WR= CAM_SENSOR_CFG_DCG_GAIN, 0x008A        // VAR= 18, 0x0026, 0x008A
```

Figure 71. Log Window in DevWare



1. Load presets from an INI file or drag selected text to the "Preset Content" window area.
2. Press "Convert" in order to view the corresponding flash records.
3. Press "Add" in order to add the translated records to the flash table.

Preset Name
TEMPORARY

```
VAR= 18, 0X0038, 0X0001
VAR= 31, 0X0000, 0X2800
REG= 0X0040, 0X8100
```

```
TYPE=create_var_set_v1
PARAMETERS={
        0x12,    0x38,    16,    0x1,      # CAM_SEN
        0x1F,    0x0,     16,    0x2800    # CMD_HAN
}


TYPE=create_reg_set
PARAMETERS={
        0x40,    0x8100   # COMMAND_REGISTER
}
```

Figure 72. Convert Commnads to Flash Records

Customer−Specific Usage

Users are allowed to reserve space with fixed addresses in Flash/EEPROM to write customer data. Users are also allowed to use any unused flash area after the flash being programmed for customer data. The former approach is recommended for writing customer data since the addresses are not impacted if the Flash/EEPROM gets reprogrammed.

The first 48 bytes are reserved for the Main TOC by the FlashTool. Therefore, users can only reserve anywhere after the first 48 bytes. FlashTool supports commands to reserve addresses for customer data, and users can choose to write customer data directly with the FlashTool or write to it using two−wire interface commands afterwards. The procedure and example are provided as follows.

- As shown in Figure 69, check *Vendor−Specific* on the bottom left, and then click on *New*.
- Choose a sub−type from the drop−down list, and write the vendor−specific command in the following format:

```
<32-bit Address>, <8-bit Value 1>, <8-bit
Value 2>, ..., <8-bit Value n>
```

NOTE: The first 4 bytes will be header, and the actual value starts 4 bytes after the address specified in the command above.

For example, if users want to reserve 2 bytes starting at 0x79A for customer data, the command will be as follows:

- 0x796, 0x12, 0x12 (write 0x1212 as customer data), or
- 0x796, 0xff, 0xff (reserve for later writes using the serial interface)

Configure SPI Bus Speed

ASX340AT defaults SPI bus at 1.6875 MHz. This is the slowest SPI bus speed that ASX340AT supports. The Host command “Config” can be used to reconfigure SPI bus to a different speed. Refer to the Host Command Interface document for details on “Config”, as well as for a list of supported SPI bus speeds in the appendix of SPI Bus Speeds. The “Config” command can be issued anytime, and takes effect immediately. Typically, inside an FCFG file, it is recommended to include a “Config” command as the first command in the Init table.

Command Sequence

The ASX340AT supports a Command Sequence Processor, which allows host commands and register writes to be executed from a command sequence record stored in NVM. In Figure 69, in the command sequence box, click on the **New** button to create a new command sequence.

Command sequences can be invoked by associating them with a GPI pin state. Set_GPI_Association is the host command to associate a GPI pin state with a Command Sequence stored in NVM. Please refer to the Host Command Interface document for details on Set_GPI_Association command. ASX340AT offers some GPIO pins for customized usage. These GPIO pins include DOUT0 − DOUT7, LINE_VALID, FRAME_VALID, DOUT_LSB0, and DOUT_LSB1.

Under certain conditions, some GPIO pins may not be used by customers. For example, if parallel output is enabled in interlaced mode, DOUT0 − DOUT7, LINE_VALID, and FRAME_VALID are all used for the parallel port. Therefore, they cannot be used for other purposes. Table 39 summarizes all the GPIO pins.

Table 39. FUNCTION TABLE

Pin (Name)	Type	Pin Group	Mapping	Alternate Function
Unassignated	–	–	GPIO0..GPIO11	–
GPIO12	Input/Output	GPIO12	GPIO12	–
–GPIO13	Input/Output	GPIO13	GPIO13	–
Unassigned		–	GPIO14, GPIO15	–
DOUT0	Input/Output	DOUT	GPIO16	Parallel Output
DOUT1	Input/Output		GPIO17	Parallel Output
DOUT2	Input/Output		GPIO18	Parallel Output
DOUT3	Input/Output		GPIO19	Parallel Output
DOUT4	Input/Output		GPIO20	Parallel Output
DOUT5	Input/Output		GPIO21	Parallel Output
DOUT6	Input/Output		GPIO22	Parallel Output
DOUT7	Input/Output		GPIO23	Parallel Output
Unassigned	–	–	GPIO24, GPIO27	–
LINE_VALID	Input/Output	FVLV	GPIO28	Line Valid
FRAME_VALID	Input/Output		GPIO29	Frame Valid
DOUT_LSB0	Input/Output	LSB0	GPIO30	LSB0
DOUT_LSB1	Input/Output	LSB1	GPIO31	LSB1

Example:

This is to associate the DOUT_LSB0 (i.e., GPIO30) pin with two command sequences:

- Invoke Command Sequence 0 when DOUT_LSB0 = 0
- Invoke Command Sequence 1 when DOUT_LSB0 = 1

This approach can be implemented to toggle between NTSC and PAL configurations for example based on Dout_LSB0 state.

Flash records are as follows:

```
TYPE=create_command
PARAMETERS={ Set_GPI_Association, 0x0, 0x1,
0x0, 0x40000000, 0x0   }
TYPE=create_command
PARAMETERS={ Set_GPI_Association, 0x1, 0x1,
0x1, 0x40000000, 0x40000000   }
```

GPIO Triggering Events with Set_GPI_Association Command

When using the Set_GPI_Association command, the following parameters need to be specified in order: Association ID, Enable, Sequence ID, GPIO Mask, GPIO State. Note the following for Set_GPI_Association and GPIO trigger:

- Association ID is from 0 to 7, i.e., a maximum of 8 associations can be active at the same time. An association needs to be enabled outside of the command sequence that's associated with it; however it can be disabled anywhere. Once an association is disabled, it can then be enabled again with the same or a different command sequence.
  If there are two associations enabled for the same GPIO pin at different states, it is recommended to always disable each association and re−enable it in its opposite command sequence. For instance, if there are two associations below:
  − Set_GPI_Association 0, 1, 0, 0x80000000, 0
  − Set_GPI_Association 1, 1, 1, 0x80000000, 0x80000000

Suppose GPIO31 is at state 1, and command sequence 1 has been executed. Now GPIO31 changes from 1 to 0, which triggers command sequence 0 to be executed. However, during this execution, GPIO monitor will not sample the pin state. Therefore, if GPIO31 then changes back to 1 before command sequence 0 finishes execution, the GPIO monitor will detect GPIO31 to be at state 1 at the end, which mistakenly assumes association 1 was just executed. This will prevent association 1 from being re−triggered, and causing the sensor to be stuck with command sequence 0 settings while GPIO31 is at state 1.

Therefore, to prevent the above situation:

− In command sequence 0, disable association 1 and then re−enable it:

- Set_GPI_Association 1, 0, 1, 0x80000000, 0x80000000
- Set_GPI_Association 1, 1, 1, 0x80000000, 0x80000000

− In command sequence 1, disable association 0 and then re−enable it:

- Set_GPI_Association 0, 0, 0, 0x80000000, 0
- Set_GPI_Association 0, 1, 0, 0x80000000, 0

- Sequence ID is the command sequence ID. There is virtually no limit on how many command sequences can be included in an FCFG file (the limit is 16.383).
- GPIO pins are state−triggered. The GPIO monitor samples the pin states every 33 ms, which cannot be changed as it is within sensor design. This means, the GPIO monitor cannot support a pin that is changing faster than half of the sampling rate (i.e. 66 ms) based on Nyquist sampling theory. In other words, for a GPIO pin to be triggered by a state change, this state needs to be active for at least 66 ms.

Program GPIO Pin to Output Pulse With Certain Width

GPIO pins can be used to output pulses for control purpose. Below is an example to output pulses (between 9 ~ 10 frames wide) either on GPIO29 (FRAME_VALID) or GPIO30 (DOUT_LSB0) depending on an input GPI value. It also outputs an indicator signal on GPIO28 (LINE_VALID).

In the Init table section, encode the following to initialize these GPIO pins:

1. GPIO_SET_ASSOCIATION (0,1,0, 0x80000000, 0x0)
2. GPIO_SET_ASSOCIATION (1,1,1, 0x80000000, 0x80000000)
3. GPIO_SET_PROP (0x10000000, 0=Owner, 1=owned by host) − Pin group FV/LV owned by host
4. GPIO_SET_PROP (0x40000000, 0, 1) − GPIO30 owned by host
5. GPIO_SET_PROP (0x10000000, 1=Direction, 0=output) − Pin group FV/LV is an output
6. GPIO_SET_PROP (0x40000000, 1, 0) − GPIO30 is an output
7. GPIO_SET_STATE (0x10000000, 0) − GPIO28 deasserted
8. GPIO_SET_STATE (0x20000000, 0) − GPIO29 deasserted
9. GPIO_SET_STATE (0x40000000, 0) − GPIO30 deasserted
10. GPIO_SET_PROP (0x10000000, 2=Enable, 1=enabled) − Pin group FV/LV is enabled (drives pin low)
11. GPIO_SET_PROP (0x40000000, 2, 1) − GPIO30 is enabled

Command Sequence 0: triggered by input GPIO31 = 0

1. GPIO_SET_STATE (0x10000000, 1) − assert GPIO28 (LINE_VALID)
2. GPIO_SET_STATE(0x20000000, 1) − assert GPIO29 (FRAME_VALID)
3. Wait For Event(SOF)
4. Wait For Event(SOF)

5. Wait For Event (SOF)
6. Wait For Event (SOF)
7. Wait For Event (SOF)
8. Wait For Event (SOF)
9. Wait For Event (SOF)
10. Wait For Event (SOF)
11. Wait For Event (SOF)
12. Wait For Event (SOF)
13. GPIO_SET_STATE (0x20000000, 0) − de−assert GPIO29 (FRAME_VALID)

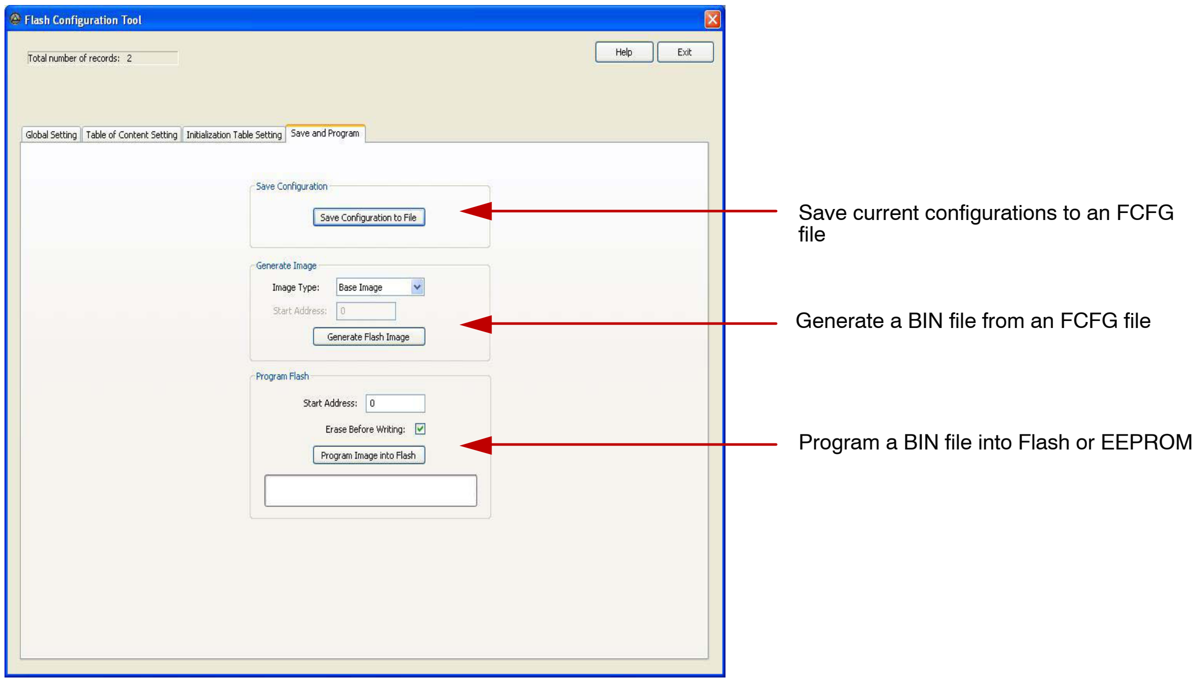
Command Sequence 1: triggered by input GPIO31 = 1

1. GPIO_SET_STATE (0x10000000, 0) − de−assert GPIO28 (LINE_VALID)
2. GPIO_SET_STATE (0x40000000, 1) − assert GPIO30 (DOUT_LSB0)
3. Wait For Event (SOF)
4. Wait For Event (SOF)
5. Wait For Event (SOF)
6. Wait For Event (SOF)
7. Wait For Event (SOF)
8. Wait For Event (SOF)
9. Wait For Event (SOF)
10. Wait For Event (SOF)
11. Wait For Event (SOF)
12. Wait For Event (SOF)
13. GPIO_SET_STATE (0x40000000, 0) − de−assert GPIO30 (DOUT_LSB0)

This will generate approximately a 150 ms (9x16.67 ms ~ 10x16.67 ms, between 9 to 10 frames for NTSC) pulse of GPIO29 and GPIO30 in each condition.

Save Flash Files and Program Flash

When finished configuring the commands on the Initialization Table Setting tab, go to the Save and Program tab (Figure 73) to save files and program flash. The settings have to be first saved to an FCFG file, then convert to a BIN file to program an NVM device.



Total number of records: 2

Image Type: Base Image

Start Address: 0

Start Address: 0

Erase Before Writing: ✓

Figure 73. Flash File Generation and Programming

Commands and register write operations are executed sequentially according to the order as shown on the *Initialization Table Setting* tab. Command sequences that are invoked based on GPIO states are executed after all the command and register write operations in the System Configuration phase are completed.

The "Start Address" used for generating a binary (BIN) file and for programming the same BIN file need to be same. This address defines where the BIN file will start to be programmed in the NVM device.

When generating a BIN file for a "Base Image" (as selected in the Image Type), the start address has to be 0. Therefore, the start address for generating and programming a base image BIN file should also be 0.

When generating a BIN file for "Image Extension", the start address needs to be set so that it does not overlap with the base image in the NVM device.

Be sure to check "Erase before Writing" when programming a base or extension image. This option will erase starting from the start address, and will only erase enough space based on the size of the BIN file to be programmed. It does not necessarily erase the whole device.

When programming the NVM with a BIN file, **onsemi**'s head board needs to be configured in flash mode (refer to

"Head Board Flash/EEPROM Configuration"). It programs the NVM device without any space starting from address 0. Any reserved addresses for customer−specific usage or unused flash area are in the state of "FFFF".If the flash has a corrupted BIN file, refer to "Boot−up Issues and Solutions".

BASE AND EXTENSION IMAGE

The BIN file layout for a Base Image can be divided into three major sections: main TOC, each individual TOC, and payload data as shown below:

Main TOC
INIT_Table TOC
CALIB_INIT TOC
PATCH TOC
…
Register Settings
Variables
Bitmaps
Strings
Patches
…

The Base fcfg file could contain empty bitmaps, strings, command sequences, etc. The generated base image BIN file will use FFFFs as the address for these empty records. The actual content of bitmaps, strings, command sequences can then be included in another fcfg file, which will be converted into an extension image. The base image, however, will need to be updated with the actual addresses for the empty records:

1. Open the base BIN file in a binary editor.
2. Refer to NVM Contents Encoding Spec document for TOC structure, and look under each individual TOC, such as overlay bitmap TOC, overlay string TOC, command seq TOC for the FFFF entries reserved for the empty records. Contact you local FAE for a copy.
3. Each empty record corresponds to one FFFF entry in its corresponding TOC. They need to be all updated.
4. Once the addresses are updated, update the checksum of the corresponding TOC record also. Refer to the NVM Contents Encoding Spec for checksum calculation.

Once the base BIN file is updated, program the base BIN file and extension BIN file sequentially. The Flash/EEPROM is now programmed with all the contents.