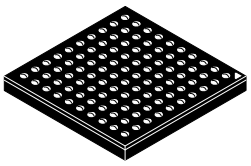
AP0100CS

AP0100CS High-Dynamic Range (HDR) Image Signal Processor (ISP)

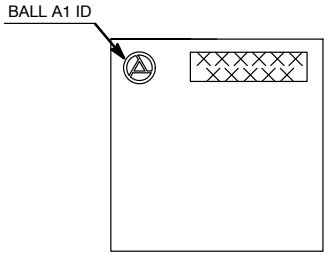
ON Semiconductor®

www.onsemi.com



VFBGA100, 7x7
CASE 138AH

MARKING DIAGRAM



XXXXXXXXXXX = Laser Marking

ORDERING INFORMATION

See detailed ordering and shipping information on page 2 of this data sheet.

General Description

The ON Semiconductor AP0100CS is a high−performance, ultra−low power in−line, digital image processor optimized for use with High Dynamic Range (HDR) sensors. The AP0100CS provides full auto−functions support (AWB and AE) and Adaptive Local Tone Mapping (ALTM) to enhance HDR images and advanced noise reduction which enables excellent low−light performance.

Table 1. KEY PERFORMANCE PARAMETERS

Parameter	Value	
Primary camera interfaces	Parallel and HiSPi	
Primary camera input	RAW12 Linear/RAW12, RAW14 (HiSPi format only) Companded	
Output interface	Analog composite, up to 16−bit parallel digital output	
Output format	YUV422 8−bit,10−bit, and 10−, 12−bit tone−mapped Bayer	
Maximum resolution	1280 × 960 (1.2 Mp)	
NTSC Output	720H × 487V	
PAL Output	720H × 576V	
Input clock range	6−30 MHz	
Supply voltage	V_{DD}IO_S	1.8 or 2.8 V nominal
	V_{DD}IO_H	2.5 or 3.3 V nominal
	V_{DD}_REG	1.8 V nominal
	V_{DD}	1.2 V nominal
	V_{DD}_PLL	1.2 V nominal
	V_{DD}_DAC	1.2 V nominal
	V_{DD}IO_OTPM	2.5 or 3.3 V nominal
	V_{DD}A_DAC	3.3 V nominal
	V_{DD}_PHY	2.8 V nominal
Operating temp.	−30°C to + 70°C	
Power consumption	185 mW	

Features

- Up to 1.2 Mp (1280 x 960) ON Semiconductor sensor support
- 45 fps at 1.2 Mp, 60 fps at 720 p
- Optimized for operation with HDR sensors
- Color and gamma correction
- Auto exposure, auto white balance, 50/60 Hz auto flicker detection and avoidance
- Adaptive Local Tone Mapping (ALTM)
- Programmable Spatial Transform Engine (STE)
- Pre−rendered Graphical Overlay
- Two−wire serial programming interface (CCIS)
- Interface to low−cost Flash or EPROM through SPI bus (to configure and load patches, etc.)
- High−level host command interface
- Standalone operation supported
- Up to 5 GPIO
- Fail−safe IO
- Multi−Camera synchronization support
- Integrated video encoder for NTSC/PAL with overlay capability and 10−bit I−DAC

Applications

- SMPTE296 HDCCTV cameras
- Surveillance network IP cameras

© Semiconductor Components Industries, LLC, 2016

ORDERING INFORMATION

Table 2. AVAILABLE PART NUMBERS

Part Number	Product Description	Orderable Product Attribute Description
AP0100CS2L00SUGA0−DR1	1Mp Co−Processor, 100−ball VFBGA	Drypack
AP0100CS2L00SPGAD3−GEVK	AP0100CS Demo Kit	
AP0100CS2L00SPGAH−GEVB	AP0100CS Head Board	

1. See the ON Semiconductor Device Nomenclature document (TND310/D) for a full description of the naming convention used for image sensors. For reference documentation, including information on evaluation kits, please visit our web site at www.onsemi.com.

FUNCTIONAL OVERVIEW

Figure 1 shows the typical configuration of the AP0100CS in a camera system. On the host side, a two−wire serial interface is used to control the operation of the AP0100CS, and image data is transferred using the analog or parallel interface between the AP0100CS and the host. The AP0100CS interface to the sensor also uses a parallel interface.

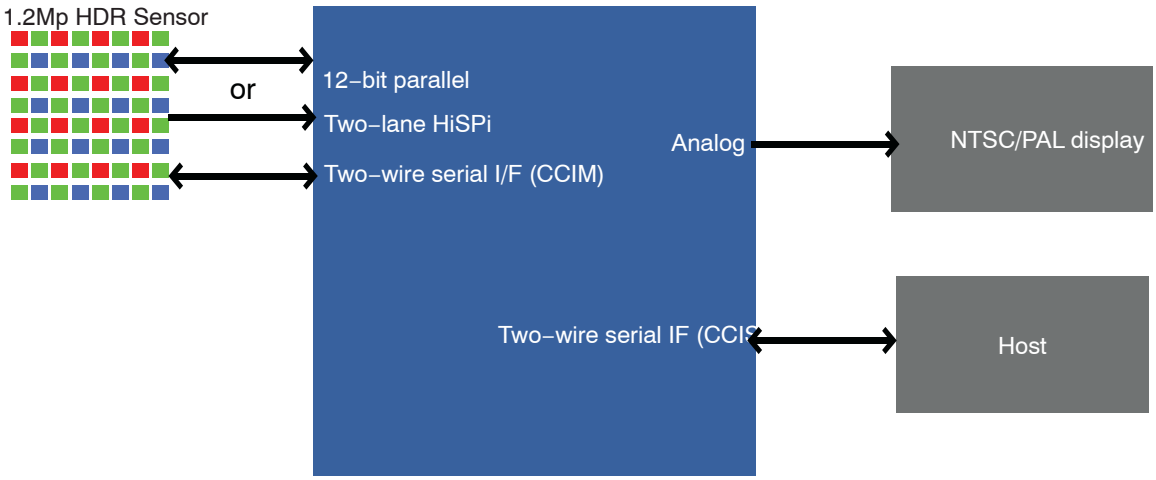


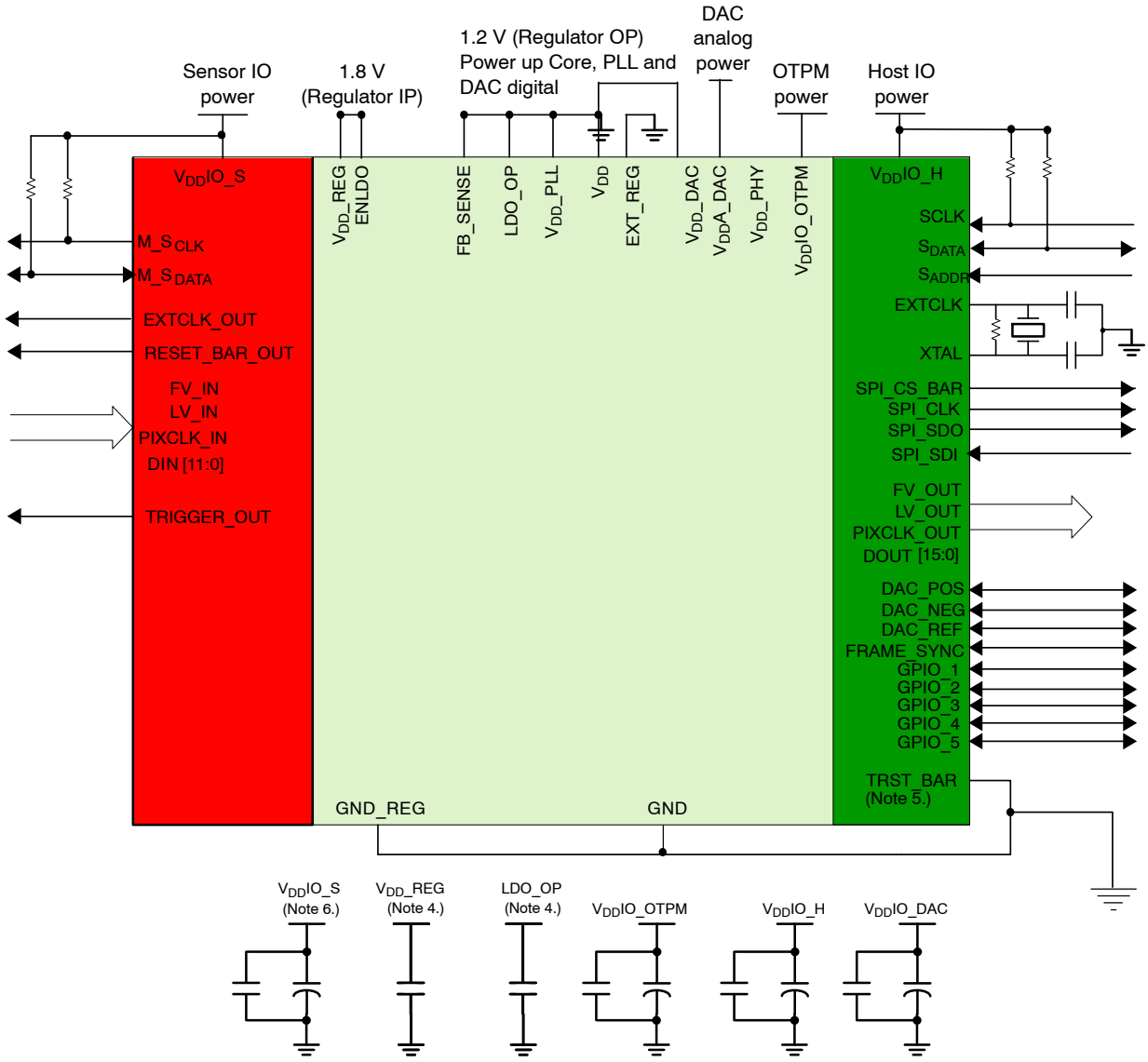
Figure 1. AP0100CS Connectivity

SYSTEM INTERFACES

Figure 2: “Typical Parallel Configuration,” on page 3 and Figure 3: “Typical HiSPi Configuration,” on page 4 show typical AP0100CS device connections.

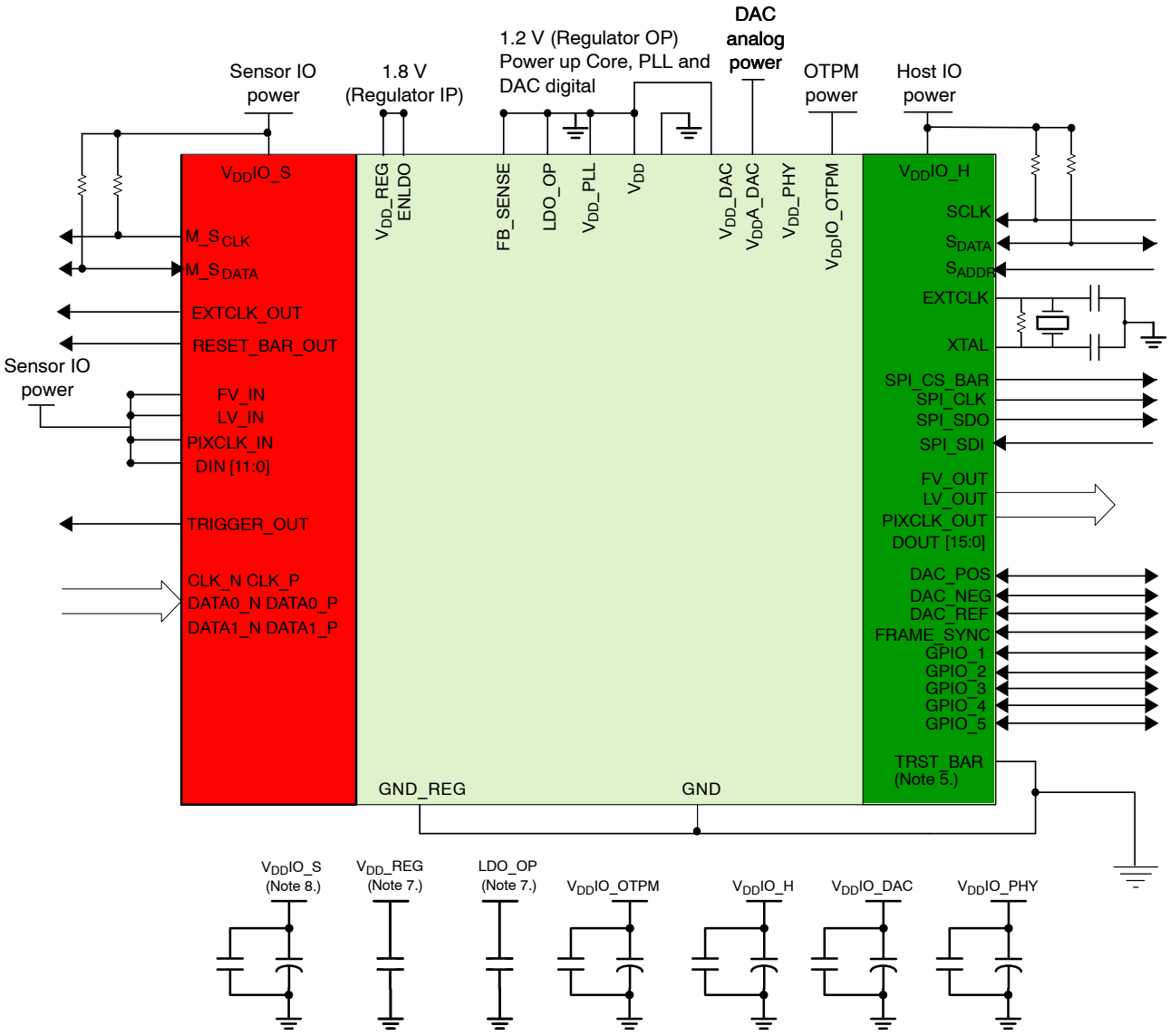
All power supply rails must be decoupled from ground using capacitors as close as possible to the package.

The AP0100CS signals to the sensor and host interfaces can be at different supply voltage levels to optimize power consumption and maximize flexibility. Table 1 on page 1 provides the signal descriptions for the AP0100CS.



NOTES: 1.. This typical configuration shows only one scenario out of multiple possible variations for this device.
2.. ON Semiconductor recommends a 1.5 kΩ resistor value for the two−wire serial interface $R_{PULL-UP}$; however, greater values may be used for slower transmission speed.
3.. RESET_BAR has an internal pull−up resistor and can be left floating if not used.
4.. The decoupling capacitors for the regulator input and output should have a value of 1.0 μF. The capacitors should be ceramic and need to have X5R or X7R dielectric.
5.. TRST_BAR connects to GND for normal operation.
6.. ON Semiconductor recommends that 0.1 μF and 1 μF decoupling capacitors for each power supply are mounted as close as possible to the pin. Actual values and numbers may vary depending on layout and design consideration.

Figure 2. Typical Parallel Configuration



NOTES: 7.. The decoupling capacitors for the regulator input and output should have a value of 1.0 μF. The capacitors should be ceramic and need to have X5R or X7R dielectric.
8.. ON Semiconductor recommends that 0.1 μF and 1 μF decoupling capacitors for each power supply are mounted as close as possible to the pin. Actual values and numbers may vary depending on layout and design consideration.

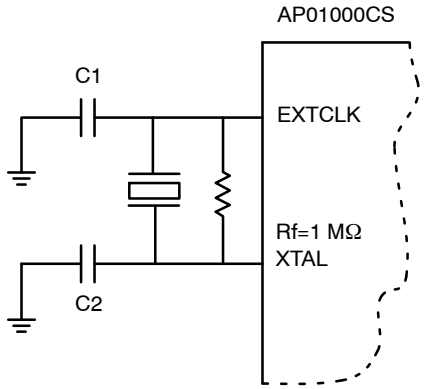
Figure 3. Typical HiSPi Configuration

HiSPi and Parallel Connection

When using the HiSPi interface, the user should connect the parallel interface to VDDIO_S. When using the parallel interface, the HiSPi interface and power supply (VDD_PHY) can be left floating.

Crystal Usage

As an alternative to using an external oscillator, a crystal may be connected between EXTCLK and XTAL. Two small loading capacitors and a feedback resistor should be added, as shown in Figure 4.



NOTE: Rf represents the feedback resistor, an Rf value of 1 MΩ would be sufficient for AP0100CS. C1 and C2 are decided according to the crystal or resonator CL specification. In the steady state of oscillation, CL is defined as (C1 x C2)/(C1+C2). In fact, the I/O ports, the bond pad, package pin and PCB traces all contribute the parasitic capacitance to C1 and C2. Therefore, CL can be rewritten to be (C1* x C2*)/(C1*+C2*), where C1*=(C1+Cin, stray) and C2*=(C2+Cout, stray). The stray capacitance for the IO ports, bond pad and package pin are known which means the formulas can be rewritten as C1*=(C1+1.5pF+Cin, PCB) and C2*=(C2+1.3pF+Cout, PCB).

Figure 4. Using a Crystal Instead of External Oscillator

Table 3. PIN DESCRIPTIONS

Name	Type	Description
EXTCLK	Input	Master input clock. This can either be a square−wave generated from an oscillator (in which case the XTAL input must be left unconnected) or direct connection to a crystal.
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.
RESET_BAR	Input/PU	Master reset signal, active LOW. This signal has an internal pull up.
S_{CLK}	Input	Two−wire serial interface clock (host interface).
S_{DATA}	I/O	Two−wire serial interface data (host interface).
S_{ADDR}	Input	Selects device address for the two−wire slave serial interface. When connected to GND, the device ID is 0x90. When wired to V_{DD}IO_H, a device ID of 0xBA is selected.
FRAME_SYNC	Input	This signal is used to synchronize to external sources or multiple cameras together. This signal should be connected to GND if not used.
STANDBY	Input	Standby mode control, active HIGH.
EXT_REG	Input	Select external regulator if tied high.
ENDLO	Input	Regulator enable (V_{DD}_REG domain).
SPI_S_{CLK}	Output	Clock output for interfacing to an external SPI flash or EEPROM memory.
SPI_SDI	Input	Data in from SPI flash or EEPROM memory. When no SPI device is fitted, this signal is used to determine whether the AP0100CS should auto−configure: 0: Do not auto−configure; two−wire interface will be used to configure the device (host−config mode) 1: Auto−configure. This signal has an internal pull−up resistor.
SPI_SDO	Output	Data out to SPI flash or EEPROM memory.
SPI_CS_BAR	Output	Chip select out to SPI flash or EEPROM memory.
EXT_CLK_OUT	Output	Clock to external sensor.
RESET_BAR_OUT	Output	Reset signal to external sensor.
M_S_{CLK}	Output	Two−wire serial interface interface clock (Master).
M_S_{DATA}	I/O	Two−wire serial interface interface clock (Master).
FV_IN	Input	Sensor frame valid input.
LV_IN	Input	Sensor line valid input.
PIXCLK_IN	Input	Sensor pixel clock output.

Table 3. PIN DESCRIPTIONS (continued)

Name	Type	Description
DIN [11:0]	Input	Sensor pixel data input D_{IN} [11:0].
CLK_N	Input	Differential HiSPi clock (sub−LVDS, negative).
CLK_P	Input	Differential HiSPi clock (sub−LVDS, positive).
DATA0_N	Input	Differential HiSPi data, lane 0 (sub−LVDS, negative).
DATA0_P	Input	Differential HiSPi data, lane 0 (sub−LVDS, positive).
DATA1_N	Input	Differential HiSPi data, lane 1 (sub−LVDS, negative).
DATA1_P	Input	Differential HiSPi data, lane 1 (sub−LVDS, positive).
TRIGGER_OUT	Output	Trigger signal for external sensor.
FV_OUT	Output	Host frame valid output (synchronous to PIXCLK_OUT).
LV_OUT	Output	Host line valid output (synchronous to PIXCLK_OUT).
PIXCLK_OUT	Output	Host pixel clock output.
DOUT [15:0]	Output	Host pixel data output (synchronous to PIXCLK_OUT) DOUT[15:0].
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.
DAC_NEG	Output	Negative video DAC output in differential mode.
DAC_REF	Output	External reference resistor for Video DAC.
GPIO [5:1]	I/O	General purpose digital I/O.
TRST_BAR	Input	Must be tied to GND in normal operation.
V_{DD}IO_S	Supply	Sensor I/O power supply.
V_{DD}IO_H	Supply	Host I/O power Supply.
V_{DD}_PLL	Supply	PLL supply.
V_{DD}	Supply	Core supply.
V_{DD}IO_OTPM	Supply	OTPM power supply.
V_{DD}_DAC	Supply	Video DAC digital power.
V_{DD}A_DAC	Supply	Video DAC analog power.
V_{DD}_PHY	Supply	PHY IO voltage for HiSPi.
GND	Supply	Ground.
V_{DD}_REG	Supply	Input to on−chip 1.8 V to 1.2 V regulator.
LDO_OP	Supply	Output from on chip 1.8 V to 1.2 V regulator.
FB_SENSE	Supply	On−chip regulator sense signal.

Table 4. PACKAGE PINOUT

	1	2	3	4	5	6	7	8	9	10
A	D_{OUT}[11]	D_{OUT}[13]	PIXCLK_ OUT	LV_OUT	GPIO_2	TRST_BAR	SPI_SDI	S_{ADDR}	S_{CLK}	STANDBY
B	D_{OUT}[12]	D_{OUT}[10]	D_{OUT}[14]	FV_OUT	GPIO_3	GPIO[5]	SPI_SCLK	S_{DATA}	TRIGGER _OUT	RESET_ BAR_OUT
C	D_{OUT}[9]	D_{OUT}[8]	D_{OUT}[15]	GPIO[1]	GPIO_4	PIXCLK _OUT	SPI_SDO	V_{DD}IO_H	M_S_{DATA}	M_S_{CLK}
D	D_{OUT}[5]	D_{OUT}[6]	D_{OUT}[7]	V_{DD}IO_H	V_{DD}IO _HOST	TRST_BAR	FRAME_ SYNC	V_{DD}	FV_IN	M_{CLK}_OUT
E	D_{OUT}[2]	D_{OUT}[3]	D_{OUT}[4]	V_{DD}IO_H	GND	GND	GND	LV_IN	PIXCLK _IN	D_{IN}[11]

Table 4. PACKAGE PINOUT (continued)

	1	2	3	4	5	6	7	8	9	10
F	$D_{OUT}[0]$	$D_{OUT}[1]$	EXTCLK	$V_{DD}IO_H$	GND	GND	GND	$V_{DD}IO_S$	$D_{IN}[9]$	$D_{IN}[10]$
G	GND	V_{DD}_PLL	XTAL	V_{DD}	V_{DD}	V_{DD}	GND	$D_{IN}[6]$	$D_{IN}[7]$	$D_{IN}[8]$
H	V_{DD}_PLL	V_{DD}_PLL	LDO_ OUTPUT	$V_{DD}IO$ _OTPM	DAC_NEG	DAC_REF	GNDA _DAC	V_{DD}_PHY	$D_{IN}[4]$	$D_{IN}[5]$
J	EXT_REG	RESET_BAR	V_{DD}_REG	V_{DD}_DAC	DAC_POS	DATA0_P	CLK_P	DATA1_N	$D_{IN}[0]$	$D_{IN}[2]$
K	GND	FB_SENSE	ENLDO	GND	$V_{DD}A_DAC$	DATA0_N	CLK_N	DATA1P_N	$D_{IN}[1]$	$D_{IN}[3]$

ON−CHIP REGULATOR

The AP0100CS has an on−chip regulator, the output from the regulator is 1.2 V and should only be used to power up the AP0100CS. It is possible to bypass the regulator and provide power to the relevant pins that need 1.2 V. Figure 5 shows how to configure the AP0100CS to bypass the internal regulator.

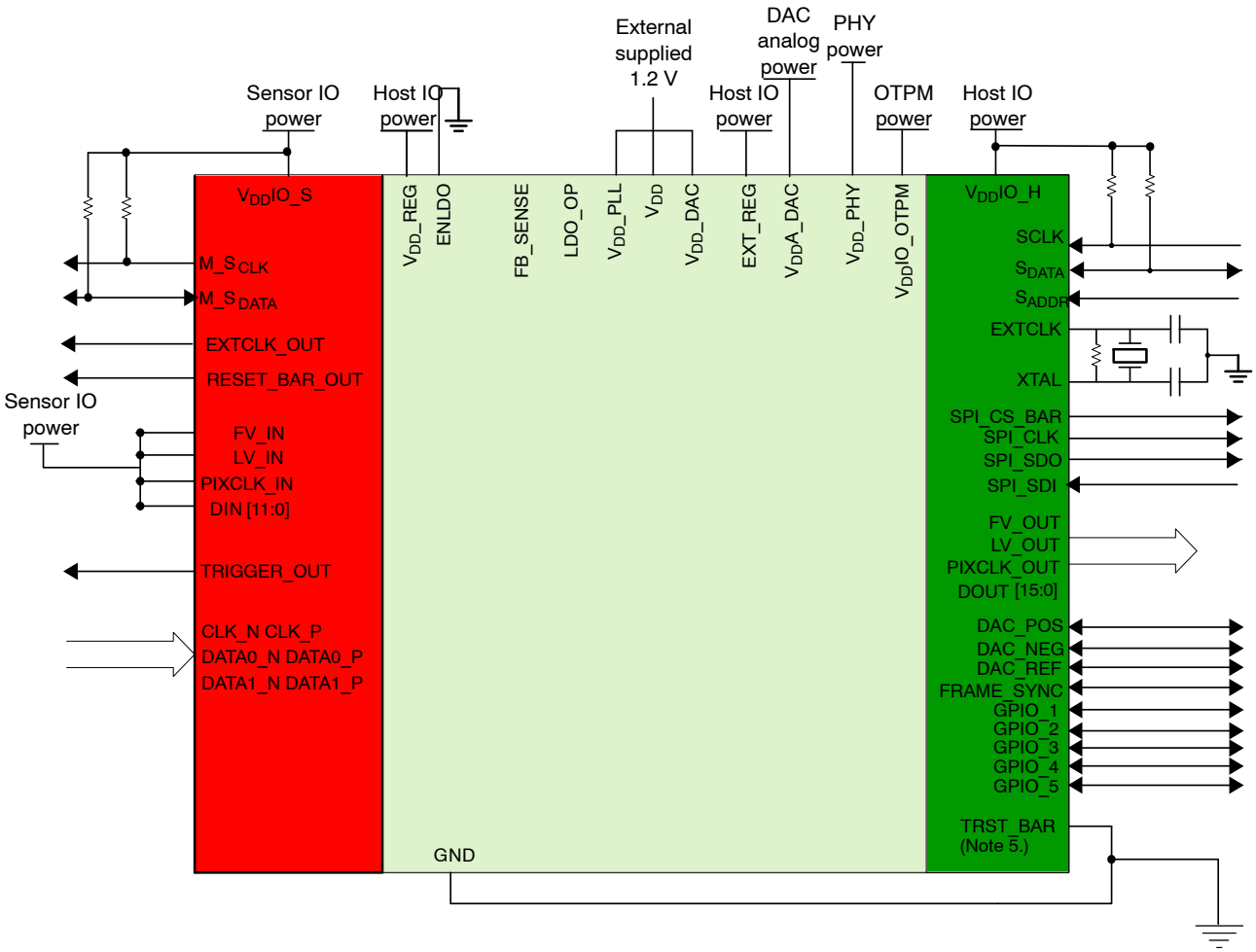


Figure 5. External Regulator

The following table summarizes the key signals when using/bypassing the regulator.

Table 5. KEY SIGNALS WHEN USING THE REGULATOR

Signal Name	Internal Regulator	External Regulator
V_{DD}_REG	1.8 V	Connect to V_{DD}IO_H
ENLDO	Connect to 1.8 V (V_{DD}_REG)	GND
FB_SENSE	1.2 V (output)	Float
LDO_OP	1.2 V (output)	Float
EXT_REG	GND	Connect to V_{DD}IO_H

Power−Up Sequence

Powering up the ISP requires voltages to be applied in a particular order, as seen in Figure 6. The timing requirements are shown in Table 6. The ISP includes a power−on reset feature that initiates a reset upon power up of the ISP.

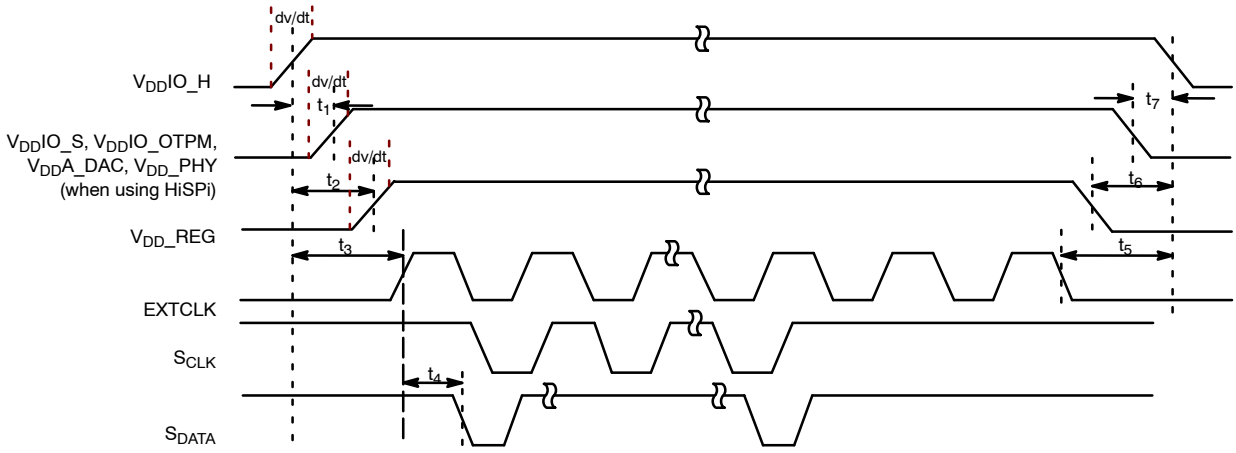


Figure 6. Power−Up and Power−Down Sequence

Table 6. POWER−UP AND POWER−DOWN SIGNAL TIMING

Symbol	Parameter	Min	Typ	Max	Unit
t1	Delay from V_{DD}IO_H to V_{DD}IO_S, V_{DD}IO_OTPM, V_{DD}A_DAC, V_{DD}_PHY (when using HiSPi)	0	−	50	ms
t2	Delay from V_{DD}IO_H to V_{DD}_REG	0	−	50	ms
t3	EXTCLK activation	t2+1	−	−	ms
t4	First serial command (Note 2)	100	−	−	EXTCLK cycles
t5	EXTCLK cutoff	t6	−	−	ms
t6	Delay from V_{DD}_REG to V_{DD}IO_H	0	−	50	ms
t7	Delay from V_{DD}IO_S, V_{DD}IO_OTPM, V_{DD}A_DAC, V_{DD}_PHY (when using HiSPi) to V_{DD}IO_H	0	−	50	ms
dv/dt	Power supply ramp time (slew rate)	−	−	0.1	V/μs

2. When using XTAL the settling time should be taken into account.

Reset

The AP0100CS has three types of reset available:

- A hard reset is issued by toggling the RESET_BAR signal.
- A soft reset is issued by writing commands through the two−wire serial interface.
- An internal power−on reset.

Table 7 on page 9 shows the output states when the part is in various states.

Table 7. OUTPUT STATES

Name	Hardware States		Firmware States				Notes
	Reset State	Default State	Hard Standby	Soft Standby	Streaming	Idle	
EXTCLK	(clock running or stopped)	(clock running)	(clock running or stopped)	(clock running)	(clock running)	(clock running)	Input
XTAL	n/a	n/a	n/a	n/a	n/a	n/a	Input
RESET_BAR	(asserted)	(negated)	(negated)	(negated)	(negated)	(negated)	Input
S_{CLK}	n/a	n/a	(clock running or stopped)	(clock running or stopped)	(clock running or stopped)	(clock running or stopped)	Input. Must always be driven to a valid logical level.
S_{DATA}	High−impedance	High−impedance	High−impedance	High−impedance	High−impedance	High−impedance	Input/Output. A valid logic level should be established by pull−up.
S_{ADDR}	n/a	n/a	n/a	n/a	n/a	n/a	Input. Must always be driven to a valid logical level.
FRAME_SYNC	n/a	n/a	n/a	n/a	n/a	n/a	Input. Must always be driven to a valid logical level.
STANDBY	n/a	(negated)	(negated)	(negated)	(negated)	(negated)	Input. Must always be driven to a valid logical level.
EXT_REG	n/a	n/a	n/a	n/a	n/a	n/a	Input. Must always be driven to a valid logical level.
ENLDO	n/a	n/a	n/a	n/a	n/a	n/a	Input. Must be tied to V_{DD}_REG or GND.
SPI_SCLK	High−impedance	driven, logic 0	driven, logic 0	driven, logic 0			Output
SPI_SDI	Internal pull−up enabled	Internal pull−up enabled	Internal pull−up enabled	Internal pull−up enabled			Input. Internal pull−up permanently enabled.
SPI_SDO	High−impedance	driven, logic 0	driven, logic 0	driven, logic 0			Output
SPI_CS_BAR	High−impedance	driven, logic 1	driven, logic 1	driven, logic 1			Output
EXT_CLK _OUT	driven, logic 0	driven, logic 0	driven, logic 0	driven, logic 0			Output
RESET_BAR _OUT	driven, logic 0	driven, logic 0	driven, logic 1	driven, logic 1			Output. Firmware will release sensor reset.
M_S_{CLK}	High−impedance	High−impedance	High−impedance	High−impedance			Input/Output. A valid logic level should be established by pull−up.
M_S_{DATA}	High−impedance	High−impedance	High−impedance	High−impedance			Input/Output. A valid logic level should be established by pull−up.
FV_IN, LV_IN, PIXCLK_IN, DIN [11:0]	n/a	n/a	n/a	n/a	Dependent on interface used	n/a	Input. Must always be driven to a valid logical level.
CLK_N	Disabled	Disabled	Dependent on interface used	Dependent on interface used	Dependent on interface used	Dependent on interface used	Input. Will be disabled and can be left floating.
CLK_P							
DATA0_N							
DATA0_P							
DATA1_N							
DATA1_P							

Table 7. OUTPUT STATES (continued)

Name	Hardware States		Firmware States				Notes
	Reset State	Default State	Hard Standby	Soft Standby	Streaming	Idle	
FV_OUT, LV_OUT, PIXCLK_OUT, DOUT [15:0]	High−impedance	Varied	Driven if used	Driven if used	Driven if used	Driven if used	Output. Default state dependent on configuration.
DAC_POS	Varied	Varied	Driven if used	Driven if used	Driven if used	Driven if used	Output. Default state dependent on configuration. Tie to ground if VDAC not used.
DAC_NEG							
DAC_REF	n/a	n/a	n/a	n/a	n/a	n/a	Input. Requires reference resistor. Tie to ground if VDAC not used.
GPIO[5:2]	High−impedance	Input, then high−impedance	Driven if used	Driven if used	Driven if used	Driven if used	Input/Output. After reset, these pins are sampled as inputs as part of auto−configuration.
GPIO1	High−impedance	High−impedance	High−impedance	High−impedance	High−impedance	High−impedance	
TRIGGER_OUT	High−impedance	High−impedance	Driven if used	Driven if used	Driven if used	Driven if used	
TRST_BAR	n/a	n/a	(negated)	(negated)	(negated)	(negated)	Input. Must always be driven to a valid logic level.

Hard Reset

The AP0100CS enters the reset state when the external RESET_BAR signal is asserted LOW, as shown in Figure 7. All the output signals will be in High−Z state.

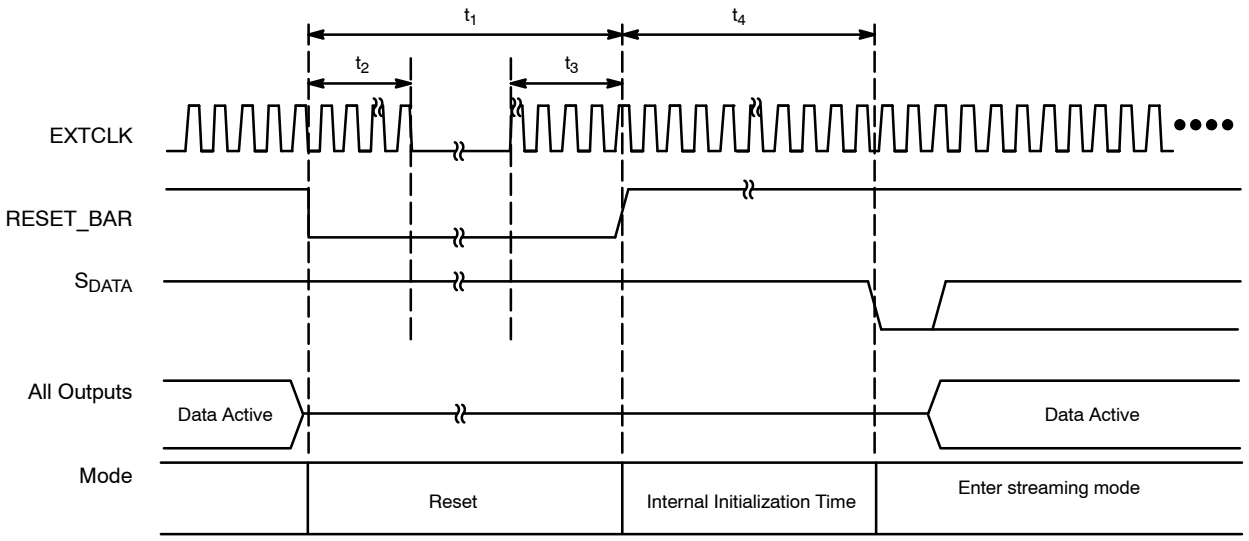


Figure 7. Hard Reset Operation

Table 8. HARD RESET

Symbol	Parameter	Min	Typ	Max	Unit
t_1	RESET_BAR pulse width	50	–	–	EXTCLK cycles
t_2	Active EXTCLK required after RESET_BAR asserted	10	–	–	
t_3	Active EXTCLK required before RESET_BAR de−asserted	10	–	–	
t_4	First two−wire serial interface communication after RESET_BAR is HIGH	100	–	–	

Soft Reset

A soft reset sequence to the AP0100CS can be activated by writing to a register through the two−wire serial interface.

Hard Standby Mode

The AP0100CS can enter hard standby mode by using the external STANDBY signal, as shown in Figure 8.

Entering Standby Mode

1. Assert STANDBY signal HIGH.

Exiting Standby Mode

1. De−assert STANDBY signal LOW.

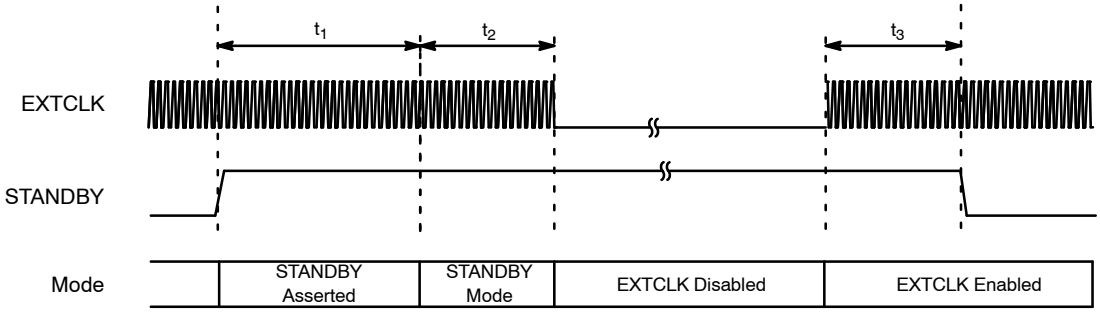


Figure 8. Hard Standby Operation

Table 9. HARD STANDBY SIGNAL TIMING

Symbol	Parameter	Min	Typ	Max	Unit
t_1	Standby entry complete	–	–	2 Frames	Lines
t_2	Active EXTCLK required after going into STANDBY mode	10	–	–	EXTCLKs
t_3	Active EXTCLK required before STANDBY de−asserted	10	–	–	EXTCLKs

MULTI−CAMERA SYNCHRONIZATION SUPPORT

The AP0100CS supports multi−camera synchronization through the FRAME_SYNC pin.

The behavior will be different depending if the user is using interlaced or progressive mode.

When using the interlaced modes, on the rising edge of FRAME_SYNC this will cause the output to stop the current frame (A) and during B the image output will be indeterminate. On the falling edge of FRAME_SYNC this will cause the re−synchronization to begin, this will continue for a period (C), during C black fields will be output. The re−synchronized interlaced signal will be available at D. During C if the user toggles the FRAME_SYNC input the AP0100CS will ignore it, the user cannot re−synchronize again until at D.

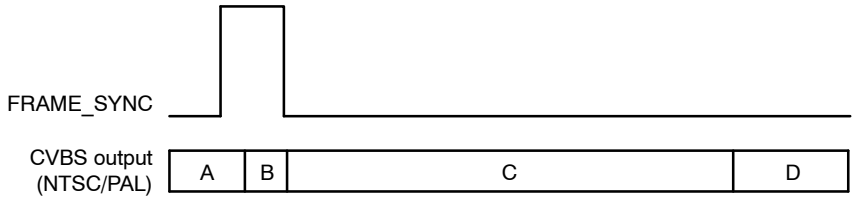
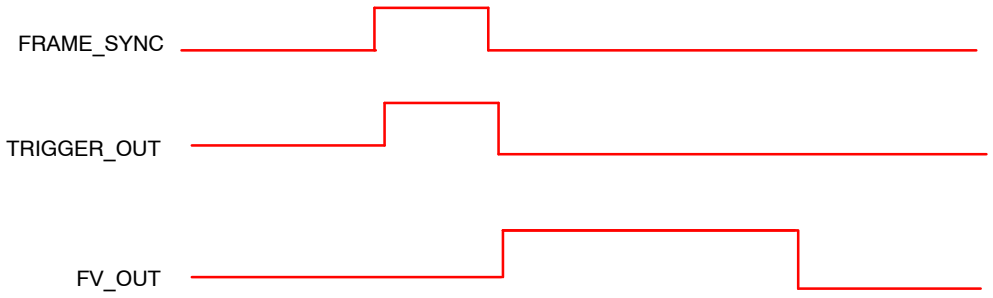


Figure 9. Frame Sync Behavior with Interlaced Mode

When using progressive mode, the host (or controlling entity) ‘broadcasts’ a sync−pulse to all cameras within the system that triggers capture. The AP0100AT will propagate the signal to the TRIGGER_OUT pin, and subsequently to the attached sensor’s TRIGGER pin.

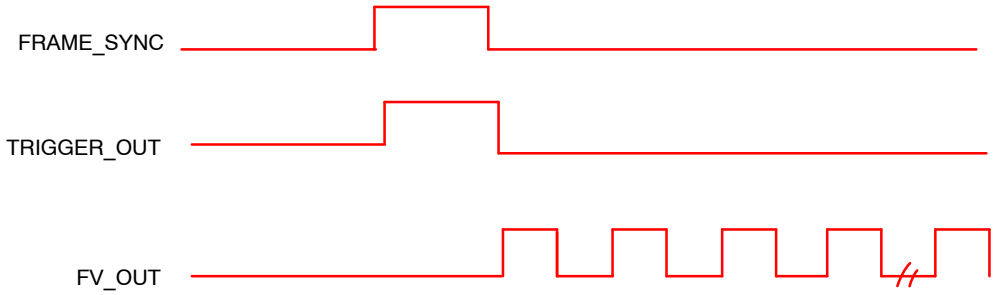
The AP0100CS supports two different trigger modes when using progressive output. The first mode supported is ‘single−shot’; this is when the trigger pulse will cause one frame to be output from the image sensor and AP0100CS (see Figure 10).



NOTE: This diagram is not to scale.

Figure 10. Single−Shot Mode

The second mode supported is called ’continuous’, this is when a trigger pulse will cause the part to continuously output frames, see Figure 11. This mode would be especially useful for applications which have multiple sensors and need to have their video streams synchronized (for example, surround view or panoramic view applications).



NOTE: This diagram is not to scale.

Figure 11. Continuous Mode

When two or more cameras have a signal applied to the FRAME_SYNC input at the same time, the respective FV_OUT signals would be synchronized within 5 PIXCLK_OUT cycles. This assumes that all cameras have the same configuration settings and that the exposure time is the same.

IMAGE FLOW PROCESSOR

Image and color processing in the AP0100CS is implemented as an image flow processor (IFP) coded in hardware logic. During normal operation, the embedded microcontroller will automatically adjust the operating parameters. For normal operation of the AP0100CS, streams of raw image data from the attached image sensor are fed into the color pipeline. The user also has the option to select a number of test patterns to be input instead of sensor data. The IFP is broken down into different sections, as outlined in Figure 12.

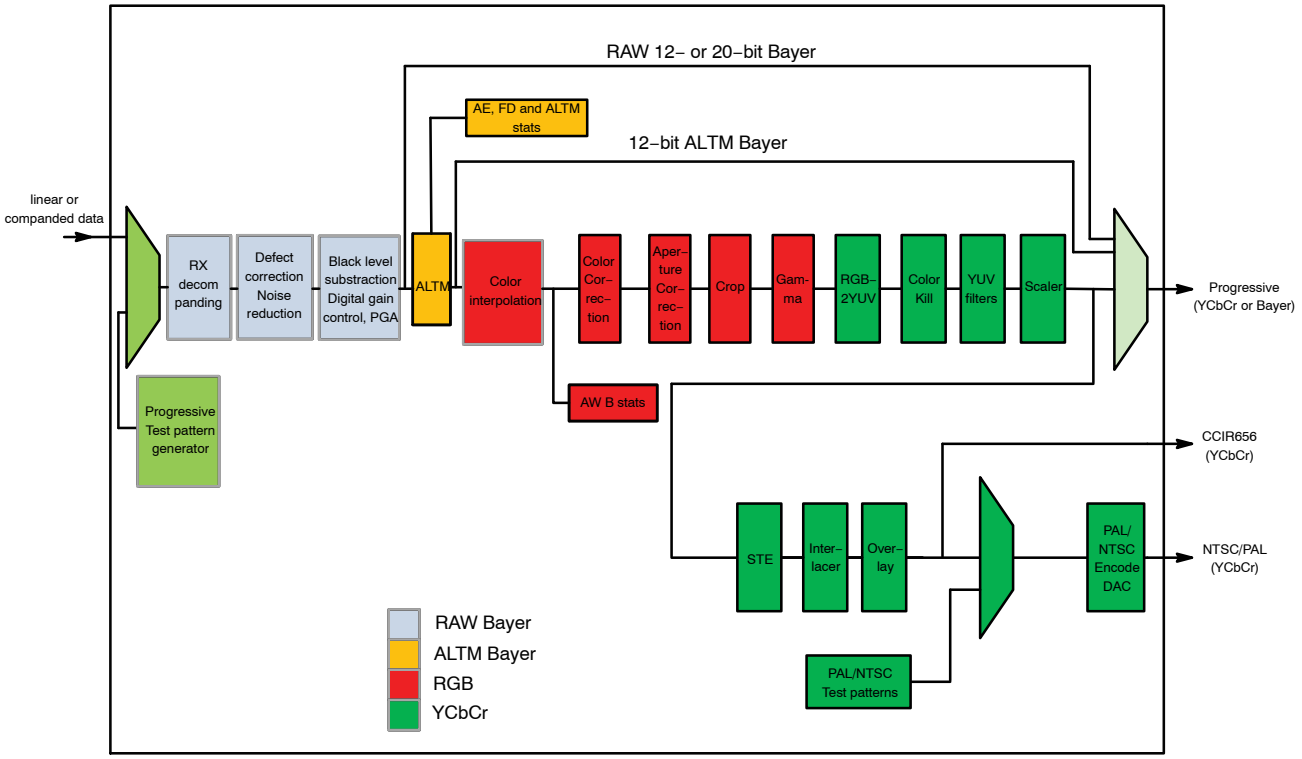


Figure 12. AP0100CS IFP

TEST PATTERNS

The AP0100CS has a number of test patterns that are available when using the progressive, NTSC and PAL modes. The test patterns can be selected by programming variables. To enter test pattern mode, set R0xC88F to 0x02 and issue a Change−Config request; to exit this mode, set R0xC88F to 0x00, and issue a Change−Config request.

NTSC and PAL test patterns can only be selected when the device is configured for interlaced operation.

Progressive Test Patterns

Table 10. PROGRESSIVE TEST PATTERNS

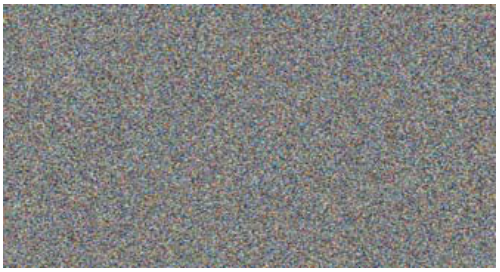
Test Pattern	Example

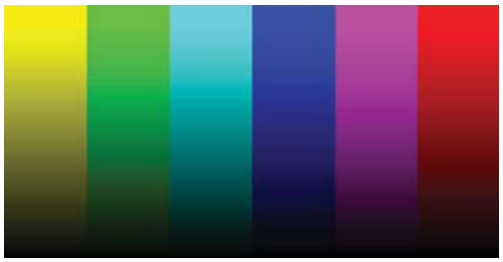
FLAT FIELD
REG= 0xC88C, 0x02 // CAM_MODE_SELECT
REG= 0xC88F, 0x01 // CAM_MODE_TEST_PATTERN_SELECT
REG= 0xC890, 0x000FFFFF // CAM_MODE_TEST_PATTERN_RED
REG= 0xC894, 0x000FFFFF // CAM_MODE_TEST_PATTERN_GREEN
REG= 0xC898, 0x000FFFFF // CAM_MODE_TEST_PATTERN_BLUE
Load = Change−Config
Changing the values in R0xC890−R0x898 will change the color of the test pattern (will require a Refresh operation).	

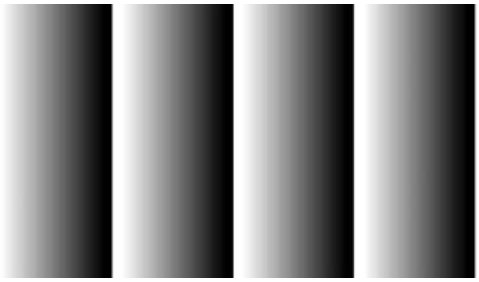
100% Color Bar
REG= 0xC88C, 0x02 // CAM_MODE_SELECT
REG= 0xC88F, 0x02 // CAM_MODE_TEST_PATTERN_SELECT
Load = Change−Config
	

Table 10. PROGRESSIVE TEST PATTERNS (continued)

Test Pattern	Example

Pseudo−Random
REG= 0xC88C, 0x02 // CAM_MODE_SELECT
REG= 0xC88F, 0x05 // CAM_MODE_TEST_PATTERN_SELECT
Load = Change−Config
	

Fade−to−Gray
REG= 0xC88C, 0x02 // CAM_MODE_SELECT
REG= 0xC88F, 0x08 // CAM_MODE_TEST_PATTERN_SELECT
Load = Change−Config
	

Linear Ramp
REG= 0xC88C, 0x02 // CAM_MODE_SELECT
REG= 0xC88F, 0x09 // CAM_MODE_TEST_PATTERN_SELECT
Load = Change−Config
	

NTSC Test Patterns

Table 11. NTSC TEST PATTERNS

Test Pattern	Example

EIA Full Field 7 Color Bars
REG= 0xC88C, 0x02 // CAM_MODE_SELECT
REG= 0xC88F, 0x14 // CAM_MODE_TEST_PATTTERN_SELECT
Load = Change−Config
	

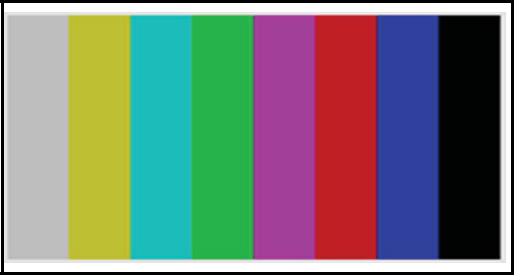
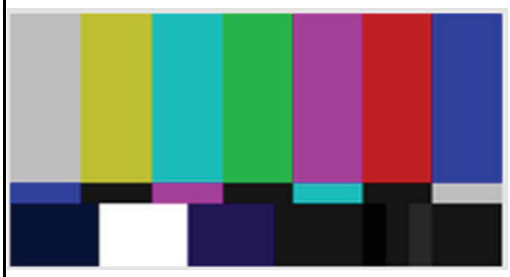
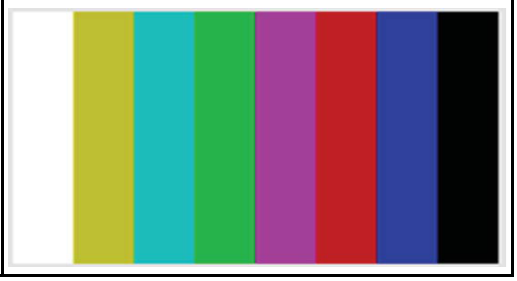
EIA Full Field 8 Color Bars
REG= 0xC88C, 0x02 // CAM_MODE_SELECT
REG= 0xC88F, 0x15 // CAM_MODE_TEST_PATTERN_SELECT
Load = Change−Config
	

Table 11. NTSC TEST PATTERNS (continued)

Test Pattern	Example

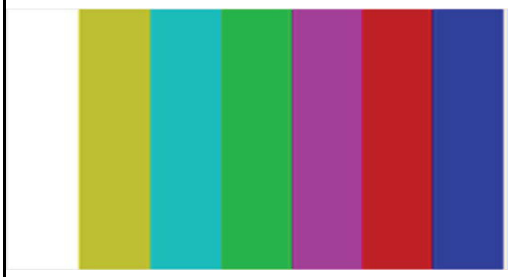
SMPTE EG 1−1990
REG= 0xC88C, 0x02 // CAM_MODE_SELECT
REG= 0xC88F, 0x16 // CAM_MODE_TEST_PATTERN_SELECT
Load = Change−Confi g
	

EIA Full Field 8 Color Bars 100 IRE
REG= 0xC88C, 0x02 // CAM_MODE_SELECT
REG= 0xC88F, 0x17 // CAM_MODE_TEST_PATTERN_SELECT
Load = Change−Config
	

PAL Test Patterns

Table 12. PAL TEST PATTERNS

Test Pattern	Example

EBU Full Field 7 Color Bars
REG= 0xC88C, 0x02 // CAM_MODE_SELECT
REG= 0xC88F, 0x1E // CAM_MODE_TEST_PATTERN_SELECT
Load = Change−Config
	

EBU Full Field 8 Color Bars
REG= 0xC88C, 0x02 // CAM_MODE_SELECT
REG= 0xC88F, 0x1F // CAM_MODE_TEST_PATTERN_SELECT
Load = Change−Config
	

Each NTSC/PAL test pattern consists of seven or eight color bars (white, yellow, cyan, green, magenta, red, blue and optionally black). The Y, Cb and Cr values for each bar are detailed in Table 13.

For the NTSC SMPTE test pattern it is also required to generate −I, +Q, −4 black and +4 black.

Table 13. PACKAGE PINOUT

	Nominal Range	White 100%	White 75%	Yellow	Cyan	Green	Magenta	Red	Blue	Black	–I	–Q	–4 black	+4 black
Y	16 to 235	235	180	162	131	112	84	65	35	16	16	16	7	25
Cb	16 to 240	128	128	44	156	72	184	100	212	128	156	171	128	128
Cr	16 to 240	128	128	142	44	58	198	212	114	128	97	148	128	128

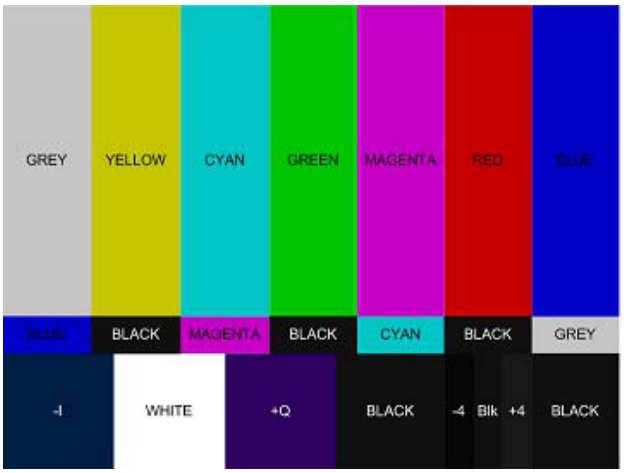


Figure 13. Test Pattern

Defect Correction

Image stream processing commences with the defect correction function immediately after data decompanding.

To obtain defect free images, the pixels marked defective during sensor readout and the pixels determined defective by the defect correction algorithms are replaced with values derived from the non−defective neighboring pixels. This image processing technique is called defect correction.

AdaCD (Adaptive Color Difference)

Automotive applications require good performance in extremely low light, even at high temperature conditions. In these stringent conditions the image sensor is prone to higher noise levels, and so efficient noise reduction techniques are required to circumvent this sensor limitation and deliver a high quality image to the user.

The AdaCD Noise Reduction Filter is able to adapt its noise filtering process to local image structure and noise level, removing most objectionable color noise while preserving edge details.

Black Level Subtraction and Digital Gain

After noise reduction, the pixel data goes through black level subtraction and multiplication of all pixel values by a programmable digital gain. Independent color channel digital gain can be adjusted with registers. Black level subtraction (to compensate for sensor data pedestal) is a single value applied to all color channels. If the black level subtraction produces a negative result for a particular pixel, the value of this pixel is set to 0.

Positional Gain Adjustments (PGA)

Lenses tend to produce images whose brightness is significantly attenuated near the edges. There are also other factors causing fixed pattern signal gradients in images captured by image sensors. The cumulative result of all these factors is known as image shading. The AP0100CS has an embedded shading correction module that can be programmed to counter the shading effects on each individual R, Gb, Gr, and B color signal.

The correction function

The correction functions can then be applied to each pixel value to equalize the response across the image as follows:

$$P_{corrected}(row, col) = P_{sensor}(row, col) \times f(row, col) \quad \text{(eq. 1)}$$

where P are the pixel values and f is the color dependent correction functions for each color channel.

Adaptive Local Tone Mapping (ALTM)

Real world scenes often have very high dynamic range (HDR) that far exceeds the electrical dynamic range of the imager. Dynamic range is defined as the luminance ratio between the brightest and the darkest object in a scene. In recent years many technologies have been developed to capture the full dynamic range of real world scenes. For example, the multiple exposure method is widely adopted for capturing high dynamic range images, which combines a series of low dynamic range images of the same scene taken under different exposure times into a single HDR image.

Even though the new digital imaging technology enables the capture of the full dynamic range, low dynamic range display devices are the limiting factor. Today's typical LCD monitor has contrast ratio around 1,000:1; however, it is not typical for an HDR image (the contrast ratio for an HDR image is around 250,000:1). Therefore, in order to reproduce HDR images on a low dynamic range display device, the captured high dynamic range must be compressed to the available range of the display device. This is commonly called tone mapping.

Tone mapping methods can be classified into global tone mapping and local tone mapping. Global tone mapping methods apply the same mapping function to all pixels. While global tone mapping methods provide computationally simple and easy to use solutions, they often cause loss of contrast and detail. A local tone mapping is thus necessary in addition to global tone mapping for the

reproduction of visually more appealing images that also reveal scene details that are important for automotive safety and surveillance applications. Local tone mapping methods use a spatially variable mapping function determined by the neighborhood of a pixel, which allows it to increase the local contrast and the visibility of some details of the image. Local methods usually yield more pleasing results because they exploit the fact that human vision is more sensitive to local contrast.

ON Semiconductor's ALTM solution significantly improves the performance over global tone mapping. ALTM is directly applied to the Bayer domain to compress the dynamic range from 20− bit to 12−bit. This allows the regular color pipeline to be used for HDR image rendering.

Color Interpolation

In the raw data stream fed by the external sensor to the IFP, each pixel is represented by a 20− or 12−bit integer number, which can be considered proportional to the pixel's response to a one−color light stimulus, red, green, or blue, depending on the pixel's position under the color filter array. Initial data processing steps, up to and including ALTM, preserve the one−color−per−pixel nature of the data stream, but after ALTM it must be converted to a three−colors−per−pixel stream appropriate for standard color processing. The conversion is done by an edge−sensitive color interpolation module. The module pads the incomplete color information available for each pixel with information extracted from an appropriate set of neighboring pixels. The algorithm used to select this set and extract the information seeks the best compromise between preserving edges and filtering out high frequency noise in flat field areas. The edge threshold can be set through register settings.

Color correction and aperture correction

To achieve good color fidelity of the IFP output, interpolated RGB values of all pixels are subjected to color correction. The IFP multiplies each vector of three pixel colors by a 3 x 3 color correction matrix. The three components of the resulting color vector are all sums of three 10−bit numbers. The color correction matrix can be either programmed by the user or automatically selected by the auto white balance (AWB) algorithm implemented in the IFP. Color correction should ideally produce output colors that are corrected for the spectral sensitivity and color crosstalk characteristics of the image sensor. The optimal values of the color correction matrix elements depend on those sensor characteristics and on the spectrum of light incident on the sensor. The color correction variables can be adjusted through register settings.

Traditionally this would have been derived from two sets of CCM, one for Warm light like Tungsten and the other for Daylight (the part would interpolate between the two matrices). This is not an optimal solution for cameras used in a Cool White Fluorescent (CWF) environment. A better solution is to provide three CCMs, which would include a matrix for CWF (interpolation now between three matrices). The AP0100CS offers this feature which will give the user improved color fidelity when under CWF type lighting.

To increase image sharpness, a programmable 2D aperture correction (sharpening filter) is applied to color−corrected image data. The gain and threshold for 2D correction can be defined through register settings.

Gamma Correction

The gamma correction curve is implemented as a piecewise linear function with 33 knee points, taking 12−bit arguments and mapping them to 10−bit output. The abscissas of the knee points are fixed at 0, 8, 16, 24, 32, 40, 48, 56, 64, 80, 96, 112, 128, 160, 192, 224, 256, 320, 384, 448, 512, 640, 768, 896, 1024, 1280, 1536, 1792, 2048, 2560, 3072, 3584, and 4096. The 10−bit ordinates are programmable through variables.

The AP0100CS has the ability to calculate the 33−point knee points based on the tuning of cam_ll_gamma and cam_ll_contrast_gradient_bright. The other method is for the host to program the 33 knee point curve themselves.

Also included in this block is a Fade−to Black curve which sets all knee points to zero and causes the image to go black in extreme low light conditions.

Color kill

To remove high−or low−light color artifacts, a color kill circuit is included. It affects only pixels whose luminance exceeds a certain preprogrammed threshold. The U and V values of those pixels are attenuated proportionally to the difference between their luminance and the threshold.

YUV color filter

As an optional processing step, noise suppression by one−dimensional low−pass filtering of Y and/or UV signals is possible. A 3− or 5−tap filter can be selected for each signal.

CAMERA CONTROL AND AUTO FUNCTIONS

Auto Exposure

The auto exposure algorithm optimizes scene exposure to minimize clipping and saturation in critical areas of the image. This is achieved by controlling exposure time and analog gains of the external sensor as well as digital gains applied to the image.

Auto exposure is implemented by a firmware algorithm that is running on the embedded microcontroller that analyzes image statistics collected by the exposure measurement engine, makes a decision, and programs the sensor and color pipeline to achieve the desired exposure. The measurement engine subdivides the image into 25 windows organized as a 5 x 5 grid

W 0,0	W 0,1	W 0,2	W 0,3	W 0,4
W 1,0	W 1,1	W 1,2	W 1,3	W 1,4
W 2,0	W 2,1	W 2,2	W 2,3	W 2,4
W 3,0	W 3,1	W 3,2	W 3,3	W 3,4
W 4,0	W 4,1	W 4,2	W 4,3	W 4,4

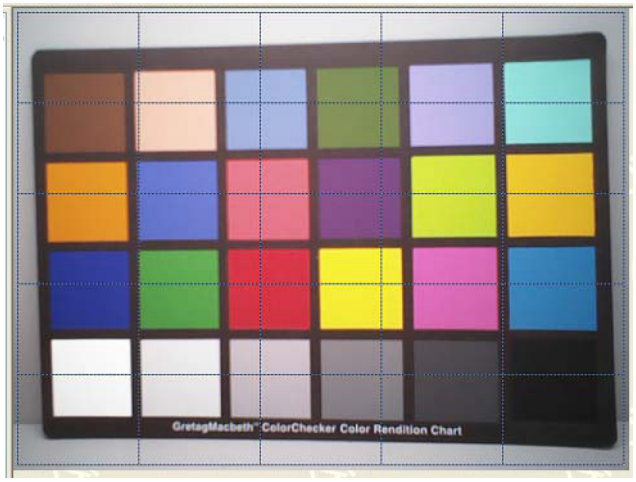


Figure 14. 5 x 5 Grid

AE Track Driver

Other algorithm features include the rejection of fast fluctuations in illumination (time averaging), control of speed of response, and control of the sensitivity to small changes. While the default settings are adequate in most situations, the user can program target brightness, measurement window, and other parameters described above.

The driver changes AE parameters (integration time, gains, and so on) to drive scene brightness to the programmable target.

To avoid unwanted reaction of AE on small fluctuations of scene brightness or momentary scene changes, the AE track driver uses a temporal filter for luma and a threshold around the AE luma target. The driver changes AE parameters only if the filtered luma is larger than the AE target step and pushes the luma beyond the threshold.

Auto White Balance

The AP0100CS has a built−in AWB algorithm 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 and a driver performing the selection of the optimal color correction matrix and IFP digital gain. While default settings of these algorithms are adequate in most situations, the user can reprogram base color correction matrices, place limits on color channel gains, and control the speed of both matrix and gain adjustments. The AP0100 CSAWB displays the current AWB position in color temperature, the range of which will be defined when programming the CCM matrixes.

The region of interest can be controlled through the combination of an inclusion window and an exclusion window.

Exposure and White Balance Control

The Sensor Manager firmware component is responsible for controlling the application of ’exposure’ and ’white balance’ within the system. This effectively means that all control of integration times and gains (whether for exposure or white balance) is delegated to the Sensor Manager. The Auto Exposure (AE) and Auto White Balance (AWB) algorithms use services provided by the Sensor Manager to apply exposure and/or white balance changes.

Dual Band IRCF

For some applications a day/night filter would be switched in/out, this option is an additional cost to the camera system. The AP0100CS supports the use of dual band IRCF, which removes the need for the switching day/night filter. Tuning support is provided for this usage case. Refer to the AP0100CS developer guide for details.

Exposure and White Balance Modes

The AP0100CS supports auto and manual exposure and white balance modes. In addition, it will operate within synchronized multi−camera systems. In this use case, one camera within the system will be the ’master’, and the others ’slaves’. The master is used to calculate the appropriate exposure and white balance. This is then applied to all slaves concurrently under host control.

Auto mode

In Auto Exposure mode the AE algorithm is responsible for calculating the appropriate exposure to keep the desired scene brightness, and for applying the exposure to the underlying hardware. In Auto White Balance mode the AWB algorithm is responsible for calculating the color temperature of the scene and applying the appropriate red and blue gains to compensate.

Triggered auto mode

The Triggered Auto Exposure and Triggered Auto White Balance modes are intended for the multicamera use cases, where a host is controlling the exposure and white balance of a number of cameras. The idea is that one camera is in triggered−auto mode (the master), and the others in hostcontrolled mode (slaves). The master camera must calculate the exposure and gains, the host then copies this to the slaves, and all changes are then applied at the same time.

Manual mode

Manual mode is intended to allow simple manual exposure and white balance control by the host. The host needs to set the CAM_AET_EXPOSURE_TIME_MS, CAM_AET_EXPOSURE_GAIN and CAM_AWB_COL−OR_TEMPERATURE controls, the camera will calculate the appropriate integration times and gains.

Host controlled

The Host Controlled mode is intended to give the host full control over exposure and gains.

FLICKER AVOIDANCE

Flicker occurs when the integration time is not an integer multiple of the period of the light intensity. The AP0100CS can be programmed to avoid flicker for 50 or 60 Hertz. For integration times below the light intensity period (10 ms for 50Hz environment), flicker cannot be avoided. The AP0100CS supports an indoor AE mode, that will ensure flicker−free operation.

FLICKER DETECTION

The AP0100CS supports flicker detection, the algorithm is designed only to detect a 50 Hz or 60 Hz flicker source.

OUTPUT FORMATTING

The pixel output data in AP0100CS will be transmitted as an 8/10 bit word over one or two clocks.

Uncompressed YCbCr Data Ordering

The AP0100CS supports swapping YCbCr mode, as illustrated in Table 14.

Table 14. YCbCr OUTPUT DATA ORDERING

Mode	Data Sequence			
Default (no swap)	Cbi	Yi	Cri	Yi+1
Swapped CrCb	Cri	Yi	Cbi	Yi+1
Swapped YC	Yi	Cbi	Yi+1	Cri
Swapped CrCb, YC	Yi	Cri	Yi+1	Cbi

The data ordering for the YCbCr output modes for AP0100CS are shown in Table 15:

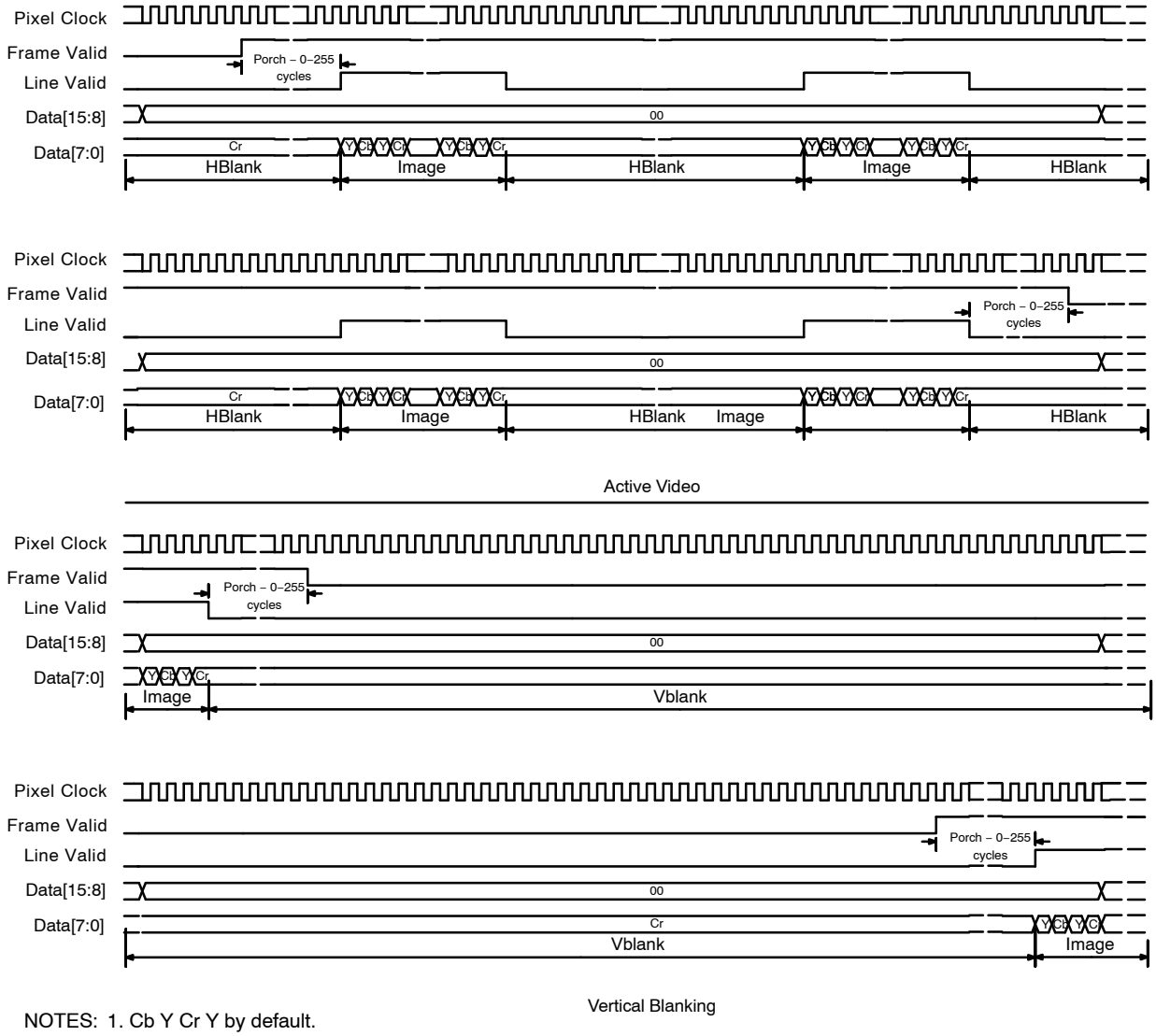
Table 15. YCbCr Output Modes (cam_port_parallel_msb_align=0x1)

Mode	Byte	Pixel i	Pixel i+1	Notes
YCbCr_422_8_8	Odd (D_{OUT} [15:8])	Cbi	Cri	Data range of 0−255 (Y = 16−235 and C = 16−240)
	Even (D_{OUT} [15:8])	Yi	Yi+1	
YCbCr_422_10_10	Odd (D_{OUT} [15:6])	Cbi	Cri	Data range of 0−1023 (Y = 64−940 and C = 64−960)
	Even (D_{OUT} [15:6])	Yi	Yi+1	
YCbCr_422_16	Single (D_{OUT} [15:0])	Cbi_Yi	Cri_Yi+1	Data range of 0−255 (Y = 16−235 and C = 16−240)

3. Odd means first cycle; even means second cycle.

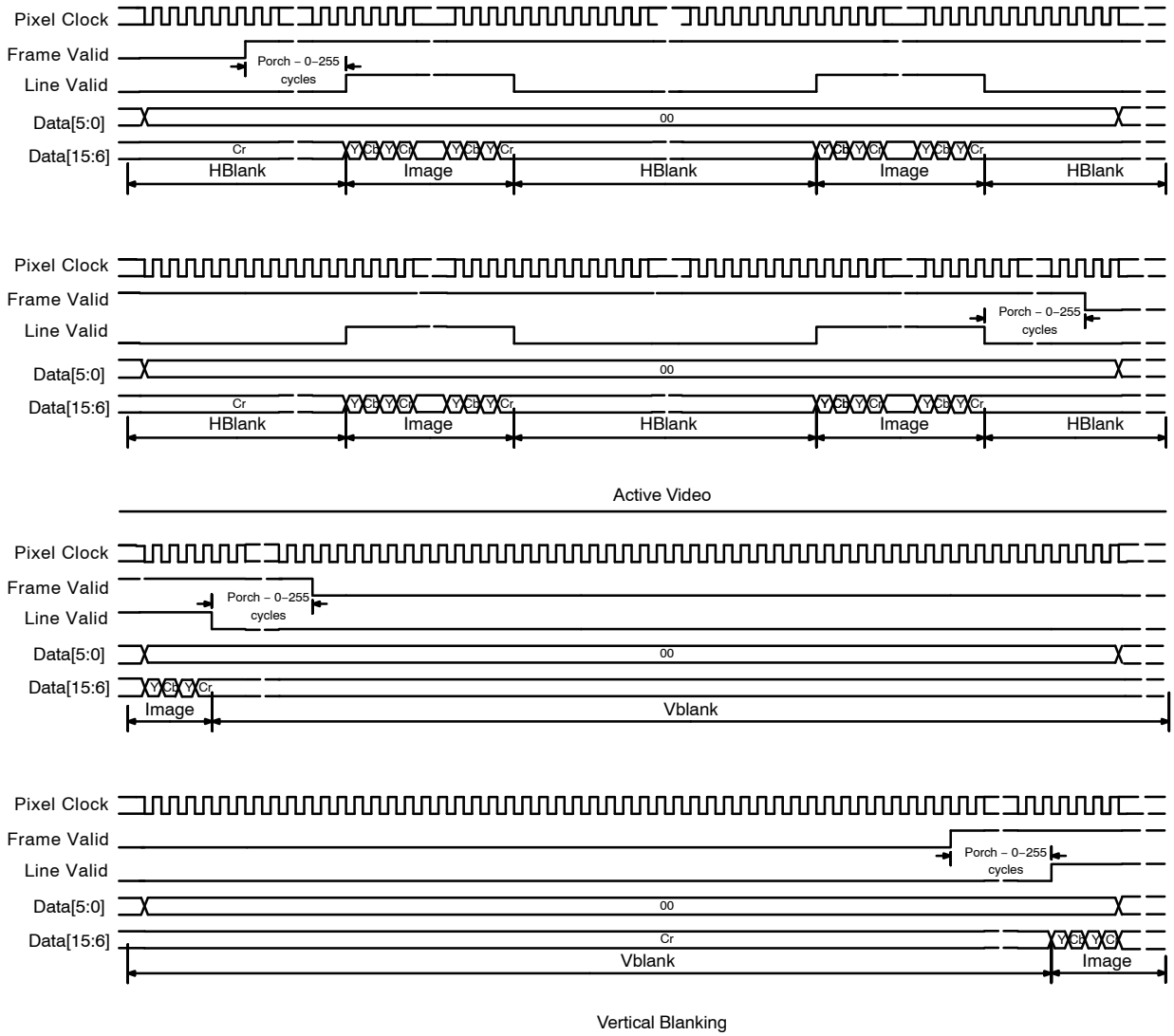
Table 16. YCbCr Output Modes (cam_port_parallel_msb_align=0x0), cam_port_parallel_swap_bytes = 0, cam_output_format_yuv_swap_red_blue = 0)

Mode	Byte	Pixel i	Pixel i+1	Notes
YCbCr_422_8_8	Odd (D_{OUT} [7:0])	Cbi	Cri	Data range of 0−255 (Y = 16−235 and C = 16−240)
	Even (D_{OUT} [7:0])	Yi	Yi+1	
YCbCr_422_10_10	Odd (D_{OUT} [9:0])	Cbi	Cri	Data range of 0−1023 (Y = 64−940 and C = 64−960)
	Even (D_{OUT} [9:0])	Yi	Yi+1	
YCbCr_422_16	Single (D_{OUT} [15:0])	Cbi_Yi	Cri_Yi+1	Data range of 0−255 (Y = 16−235 and C = 16−240)



NOTES: 1. Cb Y Cr Y by default.
2. cam_port_parallel_msb_align = 0x0

Figure 15. 8−bit YCbCr Output (YCbCr_422_8_8)



NOTES: 1. Cb Y Cr Y by default.
2. cam_port_parallel_msb_align = 0x0

Figure 16. 10−bit YCbCr Output (YCbCr_422_10_10)

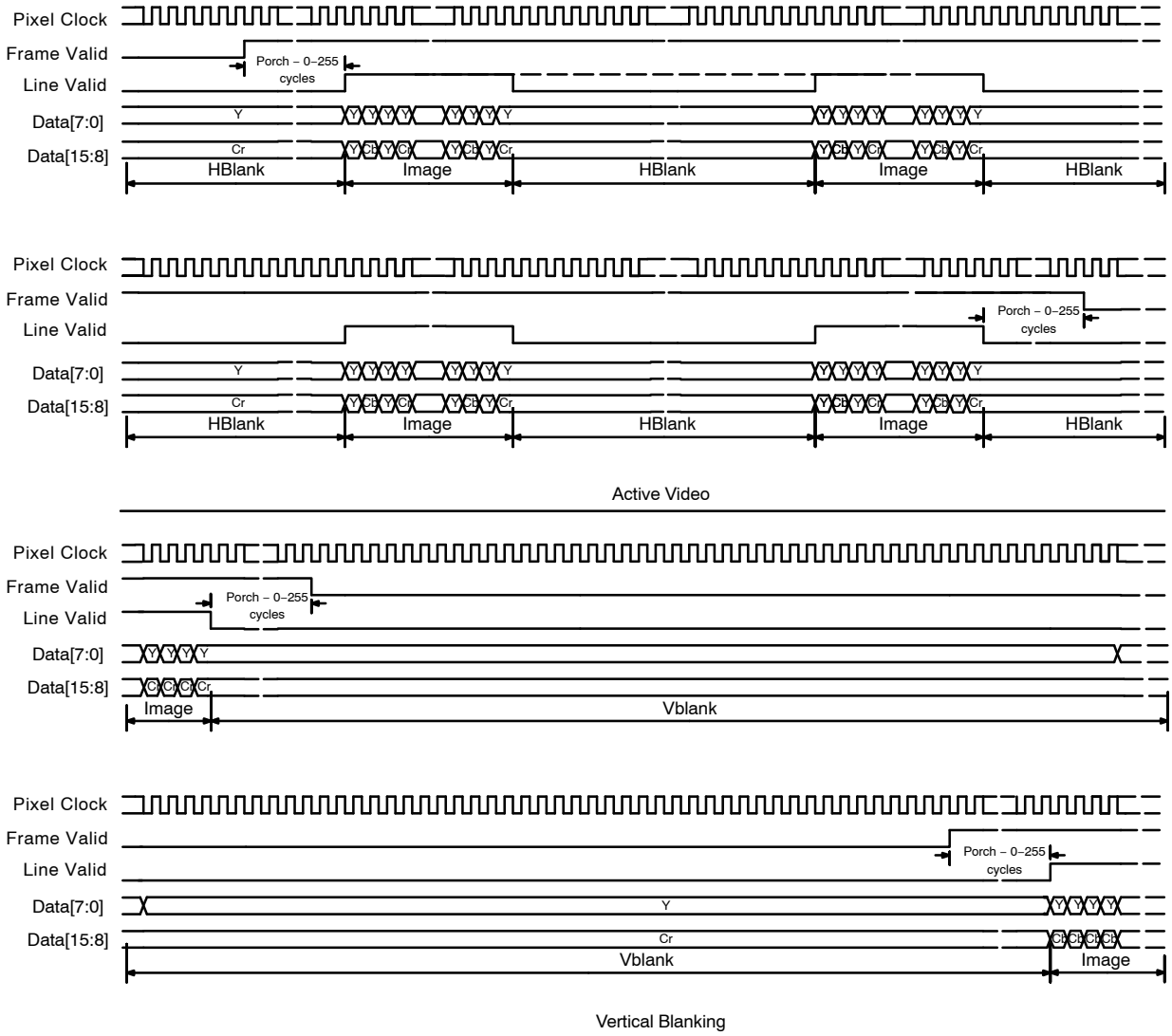


Figure 17. 16−bit YCbCr Output (YCbCr_422_16)

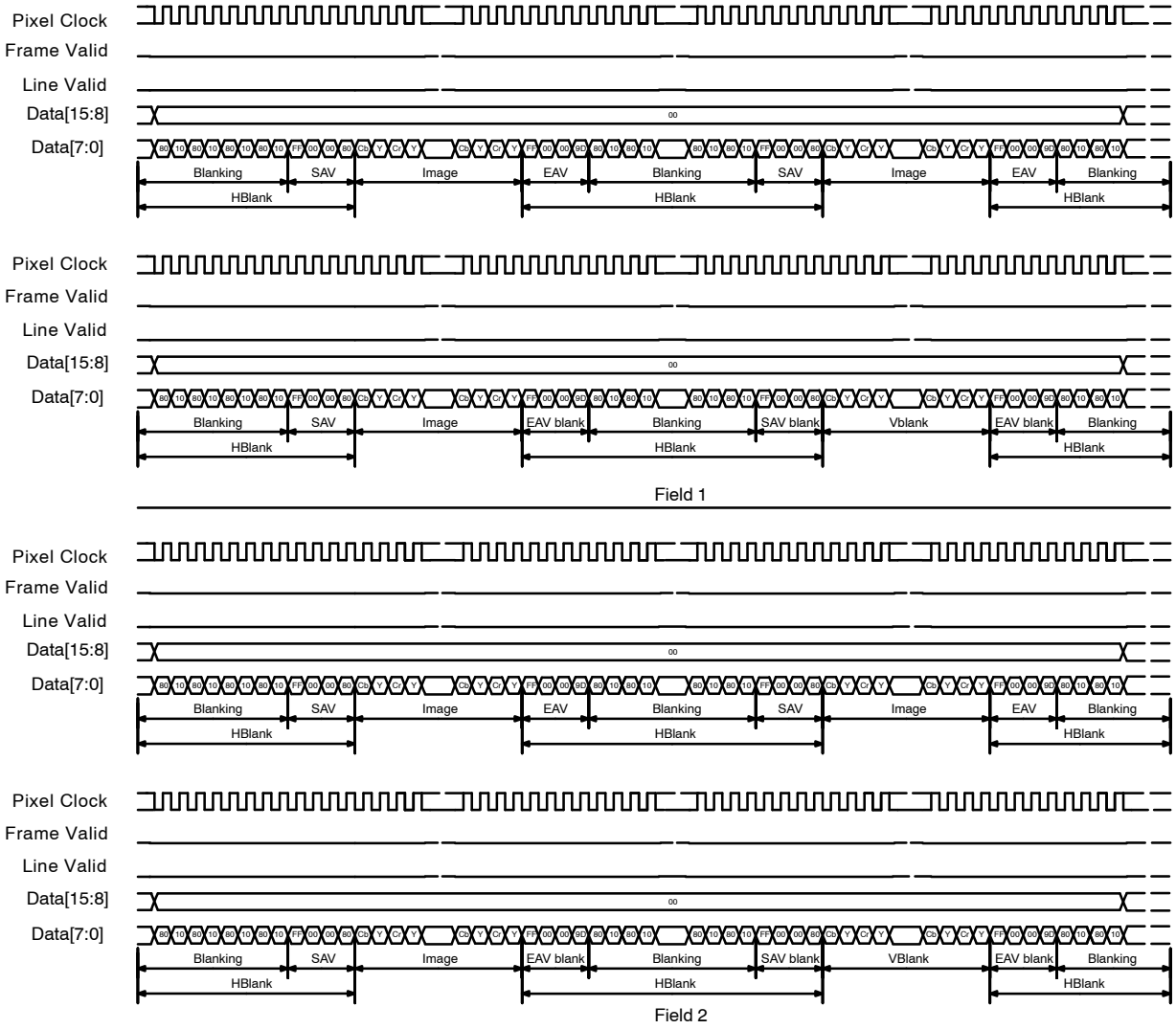


Figure 18. Typical CCIR656 Output

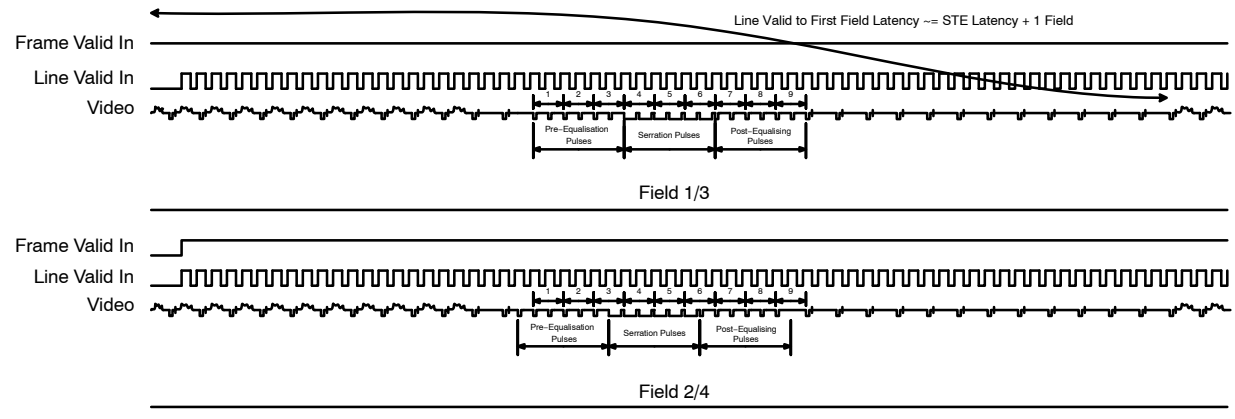


Figure 19. Typical CVBS Output (NTSC/PAL)

BAYER MODES

Bayer output modes are only available in progressive output mode before STE. The data ordering for the ALTM Bayer output modes for AP0100CS are shown in Table 17.

Table 17. ALTM BAYER OUTPUT MODES

Mode	Byte	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
ALTM_Bayer_10	Single	0	0	0	0	0	0	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
ALTM_Bayer_12	Single	0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Table 17 and Table 18 show LSB aligned data; it is possible by using register setting to obtain MSB aligned data.

The data ordering for the Bayer output modes for AP0100CS are shown in Table 18.

Table 18. BAYER OUTPUT MODES

Mode	Byte	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	Notes
Bayer_12	Single	0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	RAW Bayer data

4. Bayer_12 can be selected by setting cam_mode_select = 0x1 and requesting a Change−Config operation.

Sensor Embedded Data

The AP0100CS is capable of passing sensor embedded data in Bayer output mode only.

The AP0100CS Statistics are available through the serial interface. Refer to the developer guide for details.

SPATIAL TRANSFORM ENGINE (STE)

A spatial transform is defined as a transform in which some pixels are in different positions within the input and output pictures. Examples include zoom, lens distortion correction, turn, and rotate. STE is a fully programmable engine which can perform spatial transforms and eliminates the need for an expensive DSP for image correction.

Lens Distortion Correction

Automotive backup cameras typically feature a wide FOV lens so that a single camera mounted above the center of the rear bumper can present the driver with a view of all potential obstacles immediately behind the full width of the vehicle. Lenses with a wide field of view typically exhibit at least a noticeable amount of barrel distortion.

Barrel distortion is caused by a reduction in object magnification the further away from the optical axis.

For the image to appear natural to the driver, the AP0100CS corrects this barrel distortion and reprocesses the image so that the resulting distortion is much smaller. This is called distortion correction. Distortion correction is the ability to digitally correct the lens barrel distortion and to provide a natural view of objects. In addition, with barrel distortion one can adjust the perspective view to enhance the visibility by virtually elevating the point of viewing objects.

Pan, Tilt, Zoom and Rotate

Using the STE it is possible to implement image transformations like Pan, Tilt, Zoom and Rotate.



Figure 20. Uncorrected Image



Figure 21. Zoomed



Figure 22. Zoom and Look Left



Figure 23. Zoom and Look Right

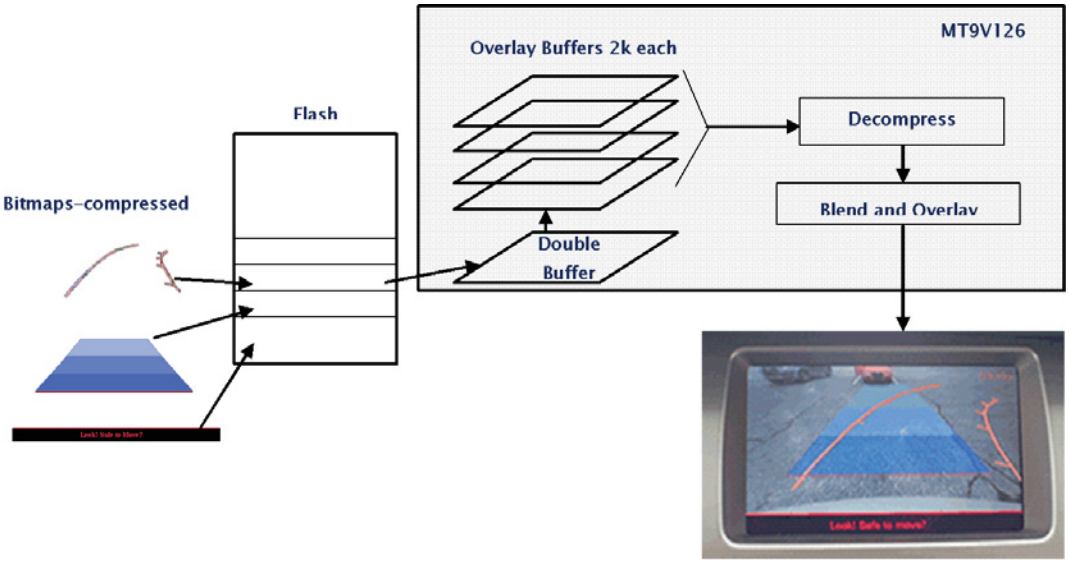
OVERLAY CAPABILITY

Figure 24 highlights the graphical overlay data flow of the AP0100CS. The images are separated to fit into 4 KB blocks of memory after compression.

- Up to seven overlays may be blended simultaneously
- Overlay size up to 720 x 576 pixels rendered
- Selectable readout: rotating order is user programmable
- Dynamic movement through predefined overlay images
- Palette of 32 colors out of 16 million with 16 colors per bitmap
- Each color has a YCbCr (8−8−8 bit) and 8 bits for the Alpha value (Transparency)
- Each layer has a built in fader which when enabled scales the Alpha value for each pixel
- Blend factors may be changed dynamically to achieve smooth transitions

The overlay engine is controlled through host commands that allow a bitmap to be written piecemeal to a memory buffer through the two−wire serial interface, and through a DMA chanel direct from SPI Flash memory. Multiple encoding passes may be required to fit an image into a 4 KB block of memory; alternatively, the image can be divided into two or more blocks to make the image fit. Every graphic image may be positioned in an x/y direction and overlap with other graphic images.

The host may load an image at any time. Under control of DMA assist, data are transferred to the off−screen buffer in compressed form. This assures that no display data are corrupted during the replenishment of the seven active overlay buffers.



NOTE: These images are not actually rendered, but show conceptual objects and object blending.

Figure 24. Overlay Data Flow

SERIAL MEMORY PARTITION

The contents of the Flash/EEPROM memory partition logically into three blocks (see Figure 25):

- Memory for overlay data and descriptors
- Memory for register settings, which may be loaded at boot−up
- Firmware extensions or software patches; in addition to the on−chip firmware, extensions reside in this block of memory

These blocks are not necessarily contiguous.

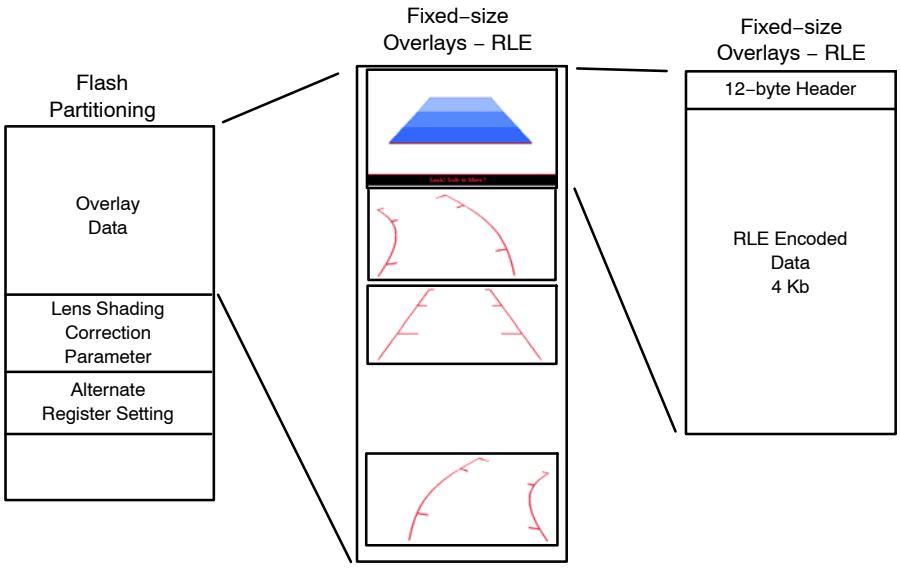


Figure 25. Memory Partitioning

OVERLAY ADJUSTMENT

To ensure a correct position of the overlay to compensate for assembly deviation, the overlay can be adjusted with assistance from the calibration statistics engine:

- The calibration statistics engine supports a windowed 8−bin luma histogram, either row−wise (vertical) or column−wise (horizontal).
- The example calibration statistics function of the firmware can be used to perform an automatic successive approximation search of a cross−hair target within the scene.
- On the first frame, the firmware performs a coarse horizontal search, followed by a coarse vertical search in the second frame.
- In subsequent frames, the firmware reduces the region−of−interest of the search to the histogram bins containing the greatest accumulator values, thereby refining the search.
- The resultant X, Y location of the cross−hair target can be used to assign a calibration value of offset selected overlay graphic image positions within the output image.
- The calibration statistics also supports a manual mode, which allows the host to access the raw accumulator values directly.

Composite Video Output

The external pin GPIO[3] can be used to configure the device for default NTSC or PAL operation. This and other video configuration settings are available as register settings accessible through the serial interface.

Single−ended and differential composite output

The composite output can be operated in a single−ended or differential mode by simply changing the external resistor configuration. For single−ended termination, see Figure 26. The differential schematic is shown in Figure 27 on page 28.

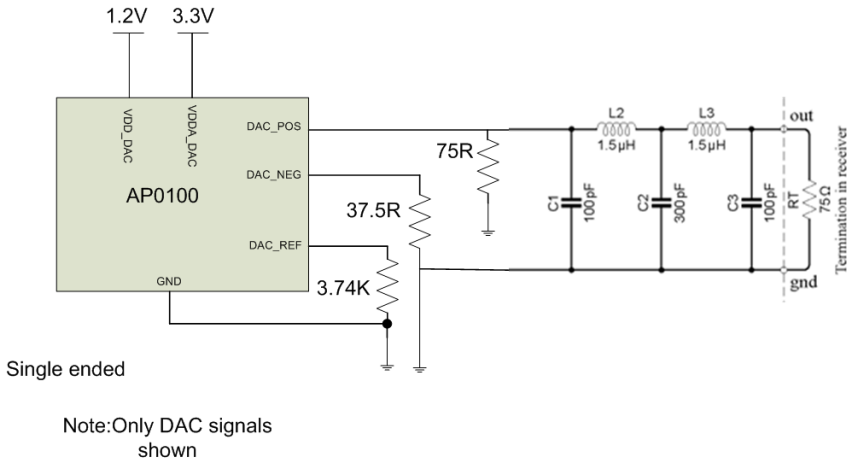


Figure 26. Single−Ended Termination

The DAC is differential, but it may be used to produce single−ended signals provided that the unused (DAC_NEG) output is terminated into a resistance to ground approximately equal to the load on the DAC_POS output. Without this termination, the internal bias circuits will not be kept in their proper operating regions and the dynamic performance of the DAC will be degraded. Termination straight into ground causes all of the power dissipation to occur on the chip, which is undesirable. If a one component saving was absolutely critical, termination straight to ground is a possibility.

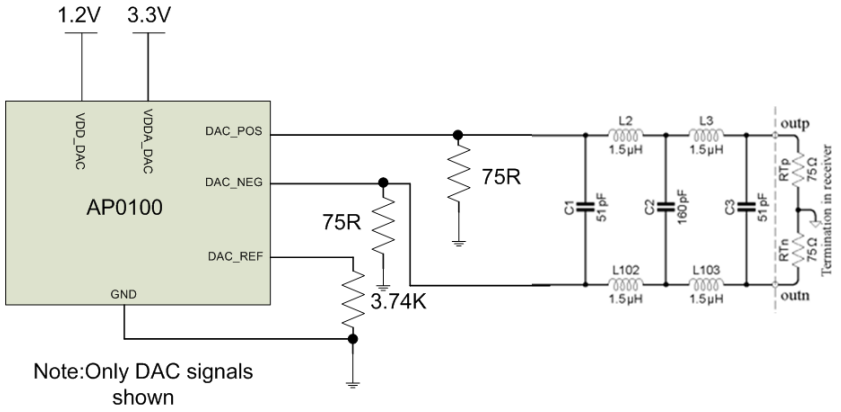
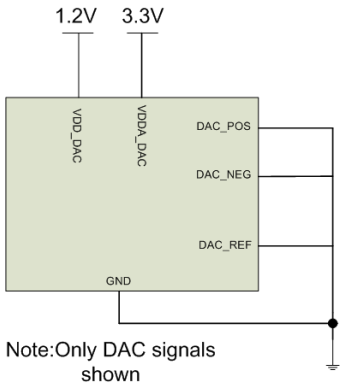


Figure 27. Differential Connection

If the user is not using the analog output then Figure 28 shows how the signals should be connected.



NOTE: Only DAC signals shown.

Figure 28. No DAC

SLAVE TWO−WIRE SERIAL INTERFACE (CCIS)

The two−wire slave serial interface bus enables read/write access to control and status registers within the AP0100CS.

The interface protocol uses a master/slave model in which a master controls one or more slave devices.

PROTOCOL

Data transfers on the two−wire serial interface bus are performed by a sequence of low−level protocol elements, as follows:

- a start or restart condition
- a slave address/data direction byte
- a 16−bit register address
- an acknowledge or a no−acknowledge bit
- data bytes
- a stop condition

The bus is idle when both SCLK and SDATA are HIGH. Control of the bus is initiated with a start condition, and the bus is released with a stop condition. Only the master can generate the start and stop conditions.

The SADDR pin is used to select between two different addresses in case of conflict with another device. If SADDR is LOW, the slave address is 0x90; if SADDR is HIGH, the slave address is 0xBA. See Table 19 below. The user can change the slave address by changing a register value.

Table 19. TWO−WIR INTERFACE ID ADDRESS SWITCHING

S_{ADDR}	Two−Wire Interface Address ID
0	0x90
1	0xBA

Start Condition

A start condition is defined as a HIGH−to−LOW transition on SDATA while SCLK is HIGH.

At the end of a transfer, the master can generate a start condition without previously generating a stop condition; this is known as a “repeated start” or “restart” condition.

Data Transfer

Data is transferred serially, 8 bits at a time, with the MSB transmitted first. Each byte of data is followed by an acknowledge bit or a no−acknowledge bit. This data transfer mechanism is used for the slave address/data direction byte and for message bytes. One data bit is transferred during

each SCLK clock period. SDATA can change when SCLK is low and must be stable while SCLK is HIGH.

Slave Address/Data Direction Byte

Bits [7:1] of this byte represent the device slave address and bit [0] indicates the data transfer direction. A “0” in bit [0] indicates a write, and a “1” indicates a read. The default slave addresses used by the AP0100CS are 0x90 (write address) and 0x91 (read address). Alternate slave addresses of 0xBA (write address) and 0xBB (read address) can be selected by asserting the SADDR input signal.

Message Byte

Message bytes are used for sending register addresses and register write data to the slave device and for retrieving register read data. The protocol used is outside the scope of the two−wire serial interface specification.

Acknowledge Bit

Each 8−bit data transfer is followed by an acknowledge bit or a no−acknowledge bit in the SCLK clock period following the data transfer. The transmitter (which is the master when writing, or the slave when reading) releases SDATA. The receiver indicates an acknowledge bit by driving SDATA LOW. As for data transfers, SDATA can change when SCLK is LOW and must be stable while SCLK is HIGH.

No−Acknowledge Bit

The no−acknowledge bit is generated when the receiver does not drive SDATA low during the SCLK clock period following a data transfer. A no−acknowledge bit is used to terminate a read sequence.

Stop Condition

A stop condition is defined as a LOW−to−HIGH transition on SDATA while SCLK is HIGH.

Typical Operation

A typical READ or WRITE sequence begins by the master generating a start condition on the bus. After the start condition, the master sends the 8−bit slave address/data direction byte. The last bit indicates whether the request is for a READ or a WRITE, where a “0” indicates a WRITE and a “1” indicates a READ. If the address matches the address of the slave device, the slave device acknowledges receipt of the address by generating an acknowledge bit on the bus.

If the request was a WRITE, the master then transfers the 16−bit register address to which a WRITE will take place. This transfer takes place as two 8−bit sequences and the slave sends an acknowledge bit after each sequence to indicate that the byte has been received. The master will then transfer the 16−bit data, as two 8−bit sequences and the slave sends an acknowledge bit after each sequence to indicate that the byte has been received. The master stops writing by generating a (re)start or stop condition. If the request was a READ, the master sends the 8−bit write slave address/data direction byte and 16−bit register address, just as in the write request. The master then generates a (re)start condition and the 8−bit read slave address/data direction byte, and clocks out the register data, 8 bits at a time. The master generates an acknowledge bit after each 8−bit transfer. The data transfer is stopped when the master sends a no−acknowledge bit.

Single READ from random location

Figure 29 shows the typical READ cycle of the host to the AP0100CS. The first two bytes sent by the host are an internal 16−bit register address. The following 2−byte READ cycle sends the contents of the registers to host.

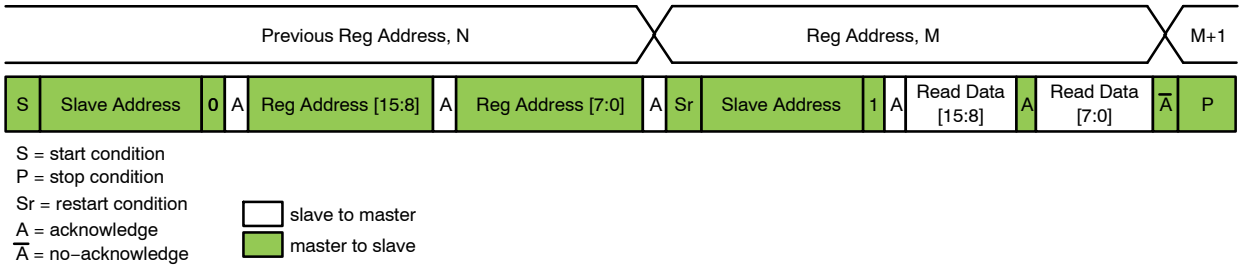


Figure 29. Single READ from Random Location

Single READ from current location

Figure 30 shows the single READ cycle without writing the address. The internal address will use the previous address value written to the register.

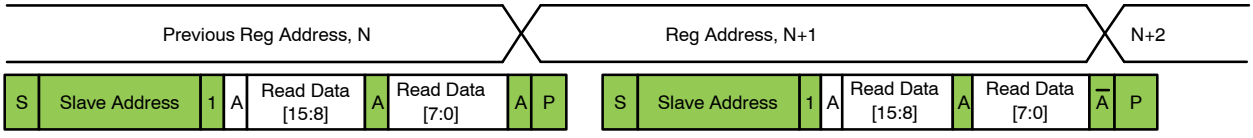


Figure 30. Single Read from Current Location

Sequential READ, start from random location

This sequence (Figure 31) starts in the same way as the single READ from random location (Figure 30 on page 29). Instead of generating a no−acknowledge bit after the first byte of data has been transferred, the master generates an acknowledge bit and continues to perform byte READs until “L” bytes have been read.

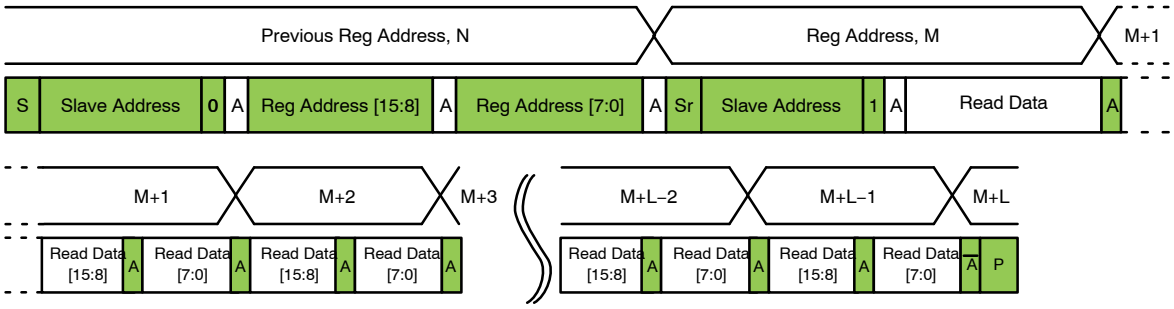


Figure 31. Sequential READ, Start from Random Location

Sequential READ, start from current location

This sequence (Figure 32) starts in the same way as the single READ from current location (Figure 30). Instead of generating a no−acknowledge bit after the first byte of data has been transferred, the master generates an acknowledge bit and continues to perform byte reads until “L” bytes have been read.

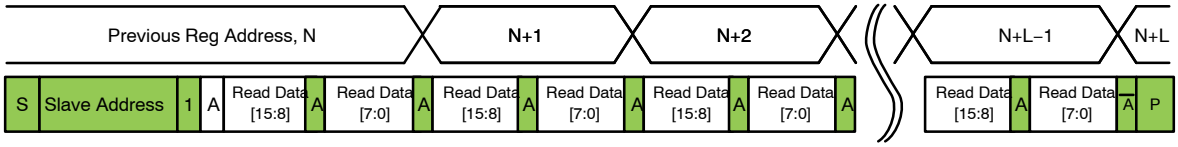


Figure 32. Sequential READ, Start from Current Location

Single write to random location

Figure 33 shows the typical WRITE cycle from the host to the AP0100CS.The first 2 bytes indicate a 16−bit address of the internal registers with most−significant byte first. The following 2 bytes indicate the 16−bit data.

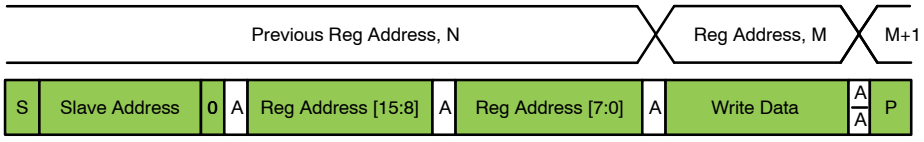


Figure 33. Single WRITE to Random Location

Sequential WRITE, start at random location

This sequence (Figure 34) starts in the same way as the single WRITE to random location (Figure 33). Instead of generating a no−acknowledge bit after the first byte of data has been transferred, the master generates an acknowledge bit and continues to perform byte writes until “L” bytes have been written. The WRITE is terminated by the master generating a stop condition.

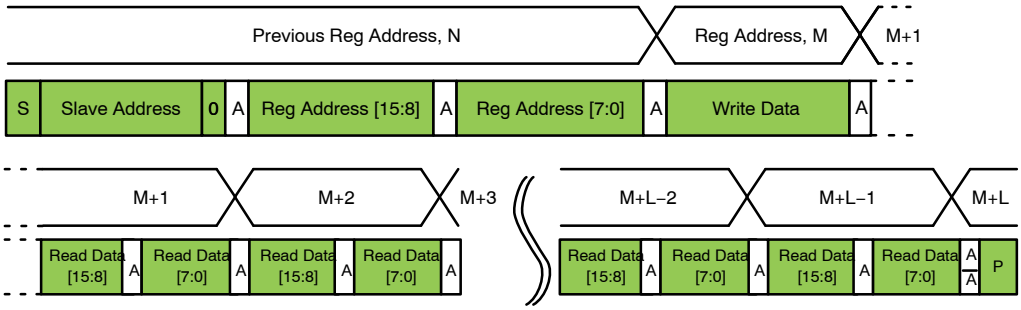


Figure 34. Sequential WRITE, Start from Current Location

Device Configuration

After power is applied and the device is out of reset (either the power on reset, hard or soft reset), it will enter a boot sequence to configure its operating mode. There are essentially three configuration modes: Flash/EEPROM Config, Auto Config, and Host Config.

The AP0100CS firmware supports a System Configuration phase at start−up. This consists of four sub−phases of execution:

Flash detection, then one of:

a. Flash Config
b. Auto Config
c. Host Config

The System Configuration phase is entered immediately following power−up or reset. Then the firmware performs Flash Detection.

Flash Detection 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 is low, then it enters the Host−Config mode.
  - If SPI_SDI is high, then it enters the Auto−Config mode.
- If a device is detected, the firmware switches to the Flash−Config mode.

In the Flash−Config mode, 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 auto−config, or to start streaming (via a Change−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 AP0100CS will take no actions until the host issues commands.

The Auto−Config mode uses the GPIO [5..2] pins to configure the operation of the device, such as video format and pedestal (see Table 21, “GPIO Bit Descriptions in Auto−Config,”). After Auto−Config completes the firmware switches to the Change−Config mode.

Table 20. SPI FLASH DEVICES

Manufacturer	Device	Type	Size	Autodetected	ManuID
Atmel	AT26DF081A	Flash	1 Mbyte	Yes	0x1f4501
Atmel	AT25DF161	Flash	2 Mbyte	Yes	0x1f4602
Sanyo (Note 5)	LE25FW806	Flash	1 Mbyte	Yes	0x622662
ST	M25P05A	Flash	64 kbyte	Yes	0x202010
ST	M25P16	Flash	2 Mbyte	Yes	0x202015
ST	M95040	EEPROM	512 byte	No	0x20ffff
ST	M95020	EEPROM	256 byte	No	0x20ffff
ST	M95010	EEPROM	128 byte	No	0x20ffff
ST	M95M01	EEPROM	128 kbyte	No	0x20ffff
Microchip	M25AA080	EEPROM	1 kbyte	No	0x29ffff
Microchip	M25LC080	EEPROM	1 kbyte	No	0x29ffff

5. Has been obsoleted.

Table 21. SPI FLASH DEVICES

	GPIO[5]	GPIO[4]	GPIO[3]	GPIO[2]
Low (“0”)	Normal	Normal	NTSC	No pedestal
High (“1”)	Vertical Flip	Horizontal mirror	PAL	Pedestal

HOST COMMAND INTERFACE

The AP0100CS has a mechanism to write higher level commands, the Host Command Interface (HCI). Once a command has been written through the HCI, it will be executed by on chip firmware and the results are reported back. EEPROM or Flash memory is also available to store commands for later execution.

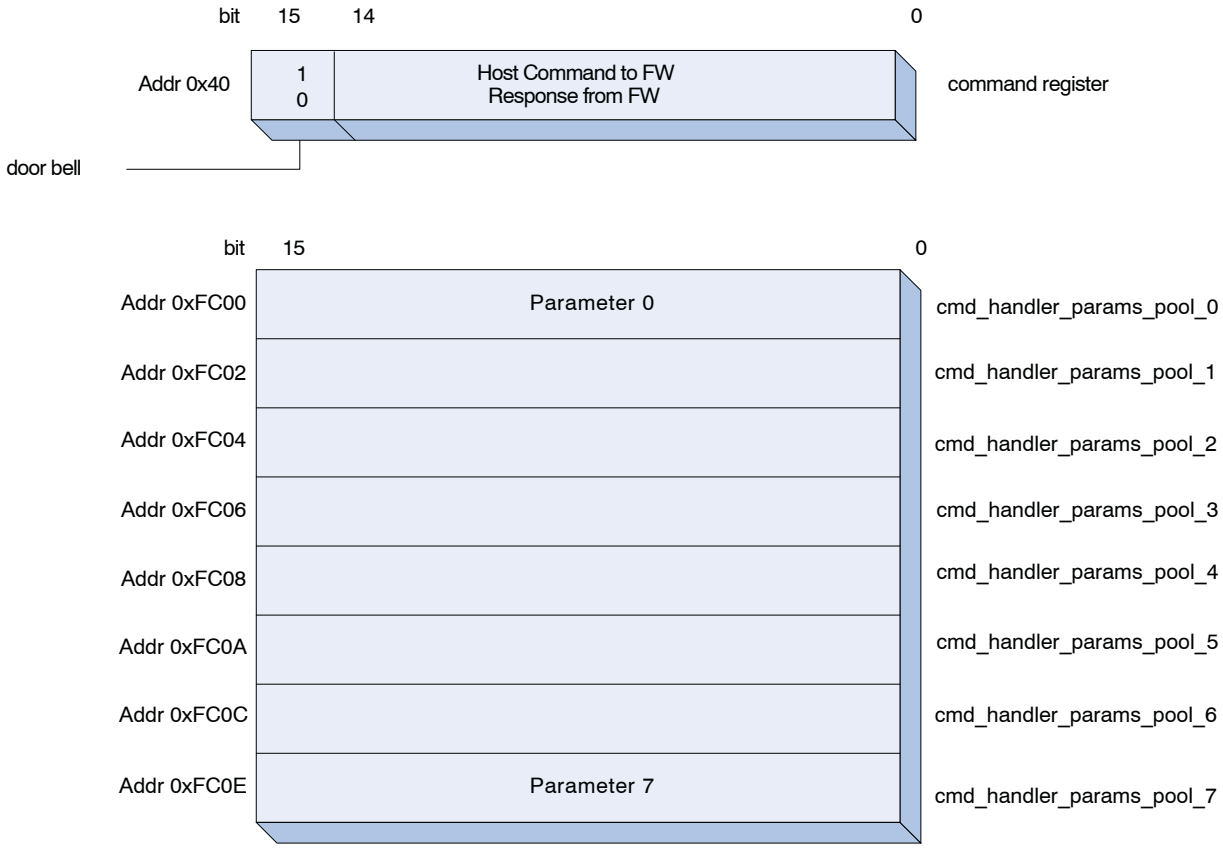


Figure 35. Interface Structure

Command Flow

The host issues a command by writing (through the two wire interface) to the Command Register. All commands are encoded with bit 15 set, which automatically generates the ‘host command’ (doorbell) interrupt to the microprocessor.

Assuming initial conditions, the host first writes the command parameters (if any) to the Parameters Pool (in the Command Handler’s shared−variable page), then writes the command to Command Register. The firmware’s interrupt handler is invoked, which immediately copies the Command Register contents. The interrupt handler then signals the Command Handler task to process the command.

If the host wishes to determine the outcome of the command, it must poll the Command Register waiting for the doorbell bit to become cleared. This indicates that the firmware completed processing the command. The contents of the Command Register indicate the command’s result status. If the command generated response parameters, the host can now retrieve these from the Parameters Pool. The host must not write to the Parameters Pool, nor issue another command, until the previous command completes. This is true even if the host does not care about the result of the previous command. It is strongly recommended that the host tests that the doorbell bit is clear before issuing a command.

Synchronous command flow

The typical ‘flow’ for synchronous commands is:

1. The host issues a ‘request’ command to perform an operation.
2. The registered command handler is invoked, validates the command parameters, then performs the operation. The handler returns the command result status to indicate the result of the operation.
3. The host retrieves the command result value, and any associated command response parameters.

Synchronous command flow

The typical ‘flow’ for asynchronous commands is:

1. The host issues a ‘request’ command to start an operation.
2. The registered command handler is invoked, validates and copies the command parameters, then signals a separate task to perform the operation. The handler returns the ENOERR return

value to indicate the command was acceptable and is in progress.

3. The host retrieves the command return value – if it is not ENOERR the host knows that the command was not accepted and is not in progress.
4. Subsequently, the host issues an appropriate ‘get status’ command to both poll whether the command has completed, and if so, retrieve any associated response parameters.
5. The registered command handler is invoked, determines the state of the command (via shared variables with the processing task), and returns either ‘EBUSY’ to indicate the command is still in progress, or it returns the result status of the command.
6. The host must re−issue the ‘get status’ command until it does not receive the EBUSY response.

Asynchronous commands exist to allow the Host to issue multiple commands to the various subsystems without having to wait for each command to complete. This prevents the host command interface from being blocked by a long−running command. Therefore, each asynchronous command has a “Get Status” (or similar) command to allow the Host to determine when the asynchronous command completes.

START−UP HOST COMMAND LOCK−OUT

The AP0100CS firmware implements an internal Host Command ‘lock’. At start−up, the firmware obtains this lock, which prevents the Host from successfully issuing a host command. All host commands will be rejected with EBUSY until the lock is freed.

The firmware releases the Host Command lock when it completes its start−up configuration processing. The time to do this is dependent upon the configuration mechanism. It is recommended that the Host poll the device with the System Manager Get State command until ENOERR is returned.

Once the host can send serial commands it should perform the following sequence.

1. POLL command_register[15] until it clears (This is called the doorbell bit).
2. Continuously issue the SYSMGR_GET_STATE command (0x8101) until the result status is not EBUSY

Below is some pseudocode that a host could use to implement the above sequence:

```
def systemWaitReadyFollowingReset(numRetries=10):
    """API function: waits for the system to be ready following reset (or powerup)
       - first wait for the doorbell bit to clear - this indicates that the device can
         accept host commands.
       - then wait until the system has completed its configuration phase; the system is
         ready when the SYSMGR_GET_STATE command does not return EBUSY.
       - note the time for the system to be ready is dependent upon the active system
         configuration mode.
       - numRetries is the number of retries before timing-out
       - returns result status code
    """

    # Wait for doorbell bit to clear (indicates device can receive host commands)
    retries = numRetries
    while (0 != retries):
        if (reg.COMMAND_REGISTER.DOORBELL.uncached_value == 0): break   # ready to receive
commands
        retries -= 1

    if (0 == retries):
        # device failed to respond in time
        return printError(ResultStatus.EIO, 'systemWaitReadyFollowingReset failed (doorbell
failed to clear)')

    # Wait for the System Manager to complete the System Configuration phase
    retries = numRetries
    while (0 != retries):

        res, currentState = sysmgrGetState()

        if (ResultStatus.ENOERR == res): break  # we're done
        if (ResultStatus.EBUSY != res):
            return printError(res, 'systemWaitReadyFollowingReset failed (sysmgrGetState
failed)')

        retries -= 1

    if (0 == retries):
        # device failed to respond in time
       return printError(ResultStatus.EAGAIN, 'systemWaitReadyFollowingReset failed (device
busy)')

    return res
```

MULTITASKING

The AP0100CS firmware is multitasking; therefore note that it is possible for an internally requested command to be in−progress when the Host issues a command. In these circumstances, the Host command is immediately rejected with EBUSY. The Host should reissue the command after a short interval.

HOST COMMANDS

Overview

The AP0100CS supports a number of functional modules or processing subsystems. Each module or subsystem exposes commands to the host to control and configure its operation.

Command Parameters

Command parameters are written to the Parameters Pool shared−variables by the host prior to invoking the command. Similarly, any Command Response parameters are also written back to the Parameters Pool by the firmware.

Result Status Codes

Table 22 shows the result status codes that are written by the Command Handler to the Host Command register, in response to a command.

Table 22. RESULT STATUS CODES

Value	Mnemonic	Typical Interpretation − each command may re−interpret
0x00	ENOERR	No error − command was successful
0x01	ENOENT	No such entity
0x02	EINTR	Operation interrupted
0x03	EIO	I/O failure
0x04	E2BIG	Too big
0x05	EBADF	Bad file/handle
0x06	EAGAIN	Would−block, try again
0x07	ENOMEM	Not enough memory/resource
0x08	EACCES	Permission denied
0x09	EBUSY	Entity busy, cannot support operation
0x0A	EEXIST	Entity exists
0x0B	ENODEV	Device not found
0x0C	EINVAL	Invalid argument
0x0D	ENOSPC	no space/resource to complete
0x0E	ERANGE	parameter out−of−range
0x0F	ENOSYS	operation not supported
0x10	EALREADY	already requested/exists

6. Any unrecognized host commands will be immediately rejected by the Command Handler, with result status code ENOSYS.

SUMMARY OF HOST COMMANDS

Table 23 on page 35 through Table 34 on page 37 show summaries of the host commands. The commands are divided into the following sections:

- System Manager
- Overlay
- GPIO
- Flash Manager
- STE
- Sequencer
- Patch Loader
- Miscellaneous
- Calibration Stats

Following is a summary of the Host Interface commands. The description gives a quick orientation. The “Type” column shows if it is an asynchronous or synchronous command. For a complete list of all commands including parameters, consult the Host Command Interface Specification document.

Table 23. SYSTEM MANAGER HOST COMMAND

System Manager Host Command	Value	Type	Description
Set state	0x8100	Asynchronous	Request the system enter a new state
Get state	0x8101	Synchronous	Get the current state of the system
Config Power Management	0x8102	Synchronous	Configures the power state of the system

Table 24. OVERLAY HOST COMMANDS

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 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)

Table 25. STE MANAGER HOST COMMANDS

STE Manager Host Commands	Value	Type	Description
Config	0x8310	Synchronous	Configure using the default NTSC or PAL configuration stored in ROM
Load Config	0x8311	Asynchronous	Load a configuration from SPI NVM to the configuration cache
Load Status	0x8312	Synchronous	Get status of a Load Config request
Write Config	0x8313	Synchronous	Write a configuration (via CCIS) to the configuration cache

Table 26. GPIO HOST COMMANDS

GPIO Host Command	Value	Type	Description
Set GPIO Property	0x8400	Synchronous	Set a property of one or more GPIO pins
Get GPIO Property	0x8401	Synchronous	Retrieve a property of a GPIO pin
Set GPIO State	0x8402	Synchronous	Set the state of a GPO pin or pins
Get GPIO State	0x8403	Synchronous	Get the state of a GPI pin or pins
Set GPI Association	0x8404	Synchronous	Associate a GPI pin state with a Command Sequence stored in SPI NVM
Get GPI Association	0x8405	Synchronous	Retrieve a GPIO pin association

Table 27. FLASH MANAGER HOST COMMAND

Flash Mgr Host Command	Value	Type	Description
Get Lock	0x8500	Asynchronous	Request the Flash Manager access lock
Lock Status	0x8501	Synchronous	Retrieve the status of the access lock request
Release Lock	0x8502	Synchronous	Release the Flash Manager access lock
Config	0x8503	Synchronous	Configure the Flash Manager and underlying SPI NVM subsystem
Read	0x8504	Asynchronous	Read data from the SPI NVM
Write	0x8505	Asynchronous	Write data to the SPI NVM
Erase Block	0x8506	Asynchronous	Erase a block of data from the SPI NVM
Erase Device	0x8507	Asynchronous	Erase the SPI NVM device
Query Device	0x8508	Asynchronous	Query device−specific information
Status	0x8509	Synchronous	Obtain status of current asynchronous operation
Config Device	0x850A	Synchronous	Configure the attached SPI NVM device

Table 28. SEQUENCER HOST COMMAND

Sequencer Host Command	Value	Type	Description
Refresh	0x8606	Asynchronous	Refresh the automatic image processing algorithm configuration
Refresh Status	0x8607	Synchronous	Retrieve the status of the last Refresh operation

Table 29. PATCH LOADER HOST COMMAND

Sequencer Host Command	Value	Type	Description
Load Patch	0x8700	Asynchronous	Load a patch from SPI NVM and automatically apply
Status	0x8701	Synchronous	Get status of an active Load Patch or Apply Patch request
Apply Patch	0x8702	Asynchronous	Apply a patch (already located in Patch RAM)
Reserve RAM	0x8706	Synchronous	Reserve RAM to contain a patch

Table 30. MISCELLANEOUS HOST COMMAND

Miscellaneous Host Command	Value	Type	Description
Invoke Command Seq	0x8900	Synchronous	Invoke a sequence of commands stored in SPI NVM
Config Command Seq Processor	0x8901	Synchronous	Configures the Command Sequence processor
Wait for Event	0x8902	Synchronous	Wait for a system event to be signalled

Table 31. CALIBRATION STATS HOST COMMANDS

Calibration Stats Host Command	Value	Type	Description
Calib Stats Control	0x8B00	Asynchronous	Start statistics gathering
Calib Stats Read	0x8B01	Synchronous	Read the results back

Table 32. EVENT MONITOR HOST COMMANDS

Event Monitor Host Command	Value	Type	Description
Event Monitor Set Association	0x8C00	Synchronous	Associate an system event with a Command Sequence stored in NVM
Event Monitor Get Association	0x8C01	Synchronous	Retrieve an event association

Table 33. CCI MANAGER HOST COMMAND

CCI Manager Host Command	Value	Type	Description
Get Lock	0x8D00	Asynchronous	Request the CCI Manager access lock
Lock Status	0x8D01	Synchronous	Retrieve the status of the access lock request
Release Lock	0x8D02	Synchronous	Release the CCI Manager access lock
Config	0x8D03	Synchronous	Release the CCI Manager access lock
Set Device	0x8D04	Synchronous	Set the target CCI device address
Read	0x8D05	Asynchronous	Read one or more bytes from a 16−bit address
Write	0x8D06	Asynchronous	Write one or more bytes to a 16−bit address
Write Bitfield	0x8D07	Asynchronous	Read−modify−write 16−bit data to a 16−bit address
CCI Status	0x8D08	Synchronous	Obtain status of current asynchronous operation

Table 34. SENSOR MANAGER HOST COMMAND

Sensor Manager Host Command	Value	Type	Description
Discover Sensor	0x8E00	Synchronous	Discover sensor
Initialize Sensor	0x8E01	Synchronous	Initialize attached sensor

USAGE MODES

How a camera based on the AP0100CS will be configured depends on what features are used. In the simplest case, an AP0100CS operating in Auto−Config mode with no customized settings might be sufficient. A back−up camera with dynamic input from the steering system will require a μC with a system bus interface. Flash sizes vary depending on the register and firmware data being transferred—somewhere between 1 KB to 16 MB. The two−wire bus is adequate since only highlevel commands are used.

In the simplest case no EEPROM or Flash memory or μC is required, as shown in Figure 36. This is truly a single chip operation.

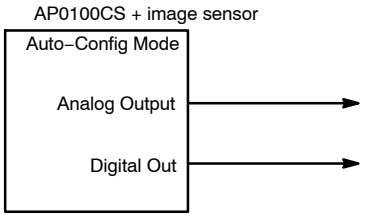


Figure 36. Auto−Config Mode

The AP0100CS can be configured by a serial EEPROM or Flash through the SPI Interface.

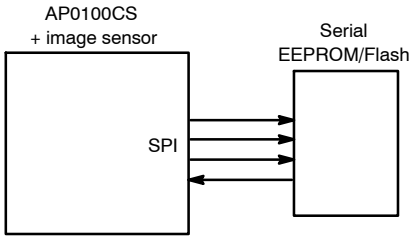


Figure 37. Flash Mode

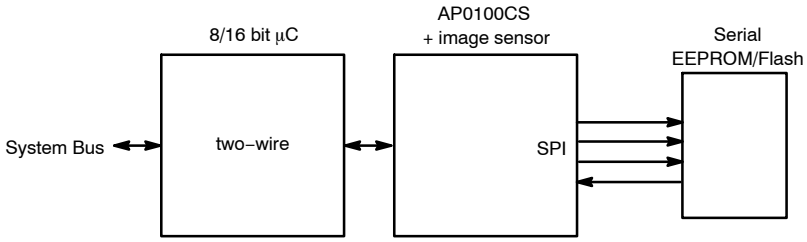


Figure 38. Host Mode with Flash

In this configuration all settings are communicated to the AP0100CS and sensor through the microcontroller.

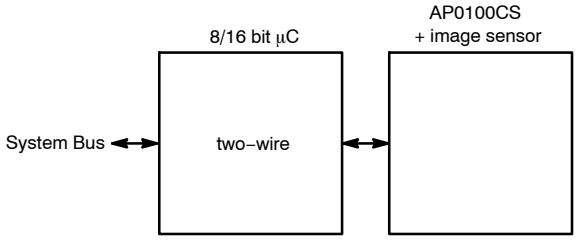


Figure 39. Host Mode

Caution: Stresses greater than those listed in Table 35 may cause permanent damage to the device. This is a stress rating only, and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect reliability.

Table 35. ABSOLUTE MAXIMUM RATINGS

Parameter	Rating		Unit
	Min	Max	
Digital power (1.8 V)	−0.3	4.95	V
Host I/O power (2.5 V, 3.3 V)	2.25	5.4	V
Sensor I/O power (1.8 V, 2.8 V)	1.7	5.4	V
Digital DAC power	1.1	2.5	V
PLL power	1.1	2.5	V
Digital core power	1.1	2.5	V
OTPM power (2.5 V, 3.3 V)	2.25	5.4	V
DC Input Voltage	−0.3	$V_{DD}IO_*$ + 0.3	V
DC Output Voltage	−0.3	$V_{DD}IO_*$ + 0.3	V
Storage temperature	−50	150	°C

Table 36. ELECTRICAL CHARACTERISTICS AND OPERATING CONDITIONS

Parameter	Min	Typ	Max	Unit
Supply input to on−chip regulator (V_{DD}_REG)	1.62	1.8	1.98	V
Host IO voltage (V_{DD}IO_H)	2.25	2.5/3.3	3.6	V
Sensor IO voltage (V_{DD}IO_S)	1.7	1.8/2.8	3.1	V
Core voltage (V_{DD})	1.08	1.2	1.32	V
PLL voltage (V_{DD}_PLL)	1.08	1.2	1.32	V
DAC digital voltage (V_{DD}_DAC)	1.08	1.2	1.32	V
DAC analog voltage (V_{DD}A_DAC)	3	3.3	3.6	V
HiSPi PHY voltage (V_{DD}_PHY)	2.3	2.8	3.1	V
OTPM power supply (V_{DD}IO_OTPM)	2.25	2.5/3.3	3.6	V
Functional operating temperature (ambient − T_A)	−30		70	°C
Storage Temperature	−55		150	°C

Table 37. AC ELECTRICAL CHARACTERISTICS

Default Setup Conditions: f_{EXTCLK}= 27 MHz, V_{DD}IO_H = V_{DD}_OTPM = 2.8 V, V_{DD}_REG = V_{DD}IO_S = 1.8 V, V_{DD}A_DAC = 3.3 V, V_{DD}_DAC = 1.2 V, T_A = 25°C unless otherwise stated

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{EXTCLK}	External clock frequency (Note 7)		6		30	MHz
t_R	External input clock rise time (Note 8)	10%−90% V_{DD}IO_H	−	2	5	ns
t_F	External input clock fall time (Note 8)	90%−10% V_{DD}IO_H	−	2	5	ns
D_{EXTCLK}	External input clock duty cycle		40	50	60	%
t_{JITTER}	External input clock jitter		−	500	−	ps
f_{PIXCLK}	Pixel clock frequency (one−clock/pixel)		6	−	74.125	MHz
	Pixel clock frequency (two−clocks/pixel)		6	−	84	MHz
$t_{RPIXCLK}$	Pixel clock rise time (10−90%)	C_{LOAD} = 35 pF	−	2	5	ns
$t_{FPIXCLK}$	Pixel clock fall time (10−90%)	C_{LOAD} = 35 pF	−	2	5	ns
t_{PD}	PIXCLK to data valid		−	1	5	ns
t_{PFH}	PIXCLK to FV HIGH		−	1	5	ns
t_{PLH}	PIXCLK to LV HIGH		−	1	5	ns
t_{PFL}	PIXCLK to FV LOW		−	1	5	ns
t_{PLL}	PIXCLK to LV LOW		−	1	5	ns

7. V_{IH}/V_{IL} restrictions apply.
8. This is applicable only a when the PLL is bypassed. When the PLL is being used then the user should ensure that V_{IH}/V_{IL} is met.

Table 38. DC ELECTRICAL CHARACTERISTICS

Symbol	Parameter	Condition	Min	Max	Unit
V_{IH}	Input HIGH voltage (Note 9)		V_{DD}IO_H or V_{DD}IO_S*0.8		V
V_{IL}	Input LOW voltage (Note 9)			V_{DD}IO_H or V_{DD}IO_S*0.2	V
I_{IN}	Input leakage current (Note 10)	V_{IN} = 0 V or V_{IN} = V_{DD}IO_H or V_{DD}IO_S		10	μA
V_{OH}	Output HIGH voltage		V_{DD}IO_H or V_{DD}IO_S*0.8		V
V_{OL}	Output LOW voltage			V_{DD}IO_H or V_{DD}IO_S*0.2	V

9. V_{IL} and V_{IH} have min/max limitations specified by absolute ratings.
10. Excludes pins that have internal PU resistors.

Table 39. VIDEO DAC ELECTRICAL CHARACTERISTICS

Default Setup Conditions: f_{EXTCLK}= 27 MHz, $V_{DD}IO_H = V_{DD}_OTPM$ = 2.8 V, $V_{DD}_REG = V_{DD}IO_S$ = 1.8 V, $V_{DD}A_DAC$ = 3.3 V, V_{DD}_DAC = 1.2 V, T_A = 25°C unless otherwise stated

Parameter	Symbol	Min	Typ	Max	Unit	Comments
DC Accuracy						
Differential Nonlinearity	DNL		+1		LSB	
Integral Nonlinearity	INL		+3		LSB	
Load Capacitance	CLOAD			10	pF	At maximum output current
Offset Error	OER			+1	% FS	For differential output only
Gain Error	DGER			+2	% FS	
Absolute Gain Error	GER			+5	% FS	

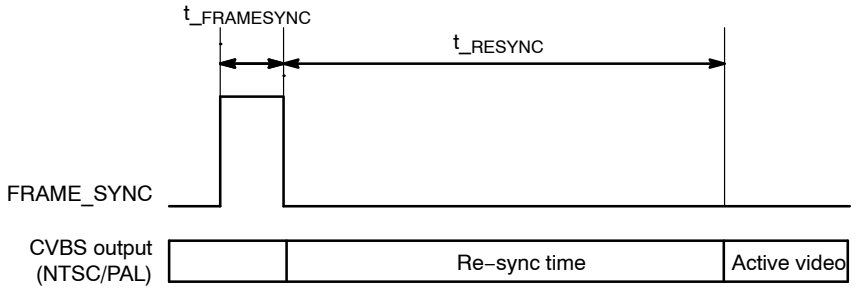


Figure 40. Frame_Sync (Interlaced Operation) Diagram

Table 40. FRAME_SYNC (INTERLACED OPERATION) PARAMETERS

Parameter	Name	Conditions	Min	Typ	Max	Unit
T_FRAME_SYNC	T_FRAME_SYNC				3	EXTCLK cycles
T_RESYNC	T_RESYNC	NTSC		100		ms
T_RESYNC	T_RESYNC	PAL		120		ms

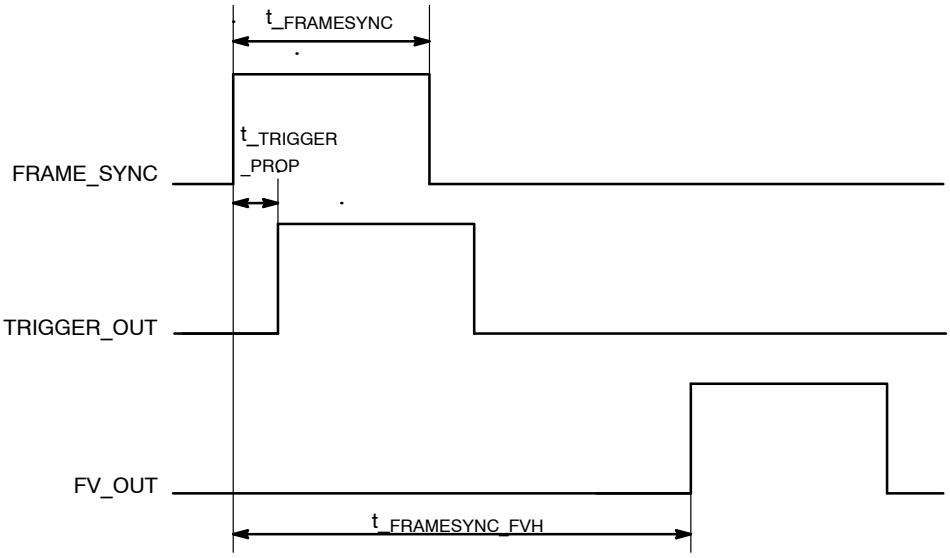


Figure 41. Frame_Sync (Progressive Operation) Diagram

Table 41. TRIGGER TIMING

Parameter	Name	Conditions	Min	Typ	Max	Unit
FRAME_SYNC to FV_OUT	$t_{FRMSYNC_FVH}$		8 lines + exposure time + sensor delay	–	–	Lines
FRAME_SYNC to TRIGGER_OUT	$t_{TRIGGER_PROP}$		–	–	9	ns
t_{FRAME_SYNC}	$t_{FRAMESYNC}$		3	–	–	EXTCLK cycles

NTSC Signal Parameters

Table 42. NTSC SIGNAL PARAMETERS

Default Setup Conditions: f_{EXTCLK}= 27 MHz, V_{DD}_REG = 1.8 V, V_{DD}_IO_S = 1.8 V, V_{DD}A_DAC = 3.3 V, V_{DD}IO_OTPM = 2.5 V, V_{DD}_PHY = 2.5 V

Parameter	Min	Typ	Max	Units
Line Frequency	15734.25	15734.27	15734.28	Hz
Field Frequency	59.94	59.94	59.94	Hz
Sync Rise Time	111	148	222	ns
Sync Fall Time	111	148	222	ns
Sync Width	4.60	4.74	4.80	μs
Sync Level (Note 12)	39	40	41	IRE
Burst Level (Note 12)	36	40	44	IRE
Sync to Setup (with pedestal off)	9.2	9.5	10.3	μs
Sync to Burst Start	4.71	5.3	5.71	μs
Front Porch (Note 11,12,13)	1.27	1.7	2.22	μs
Black Level (Note 11,12,13)	5	7.5	10	IRE
White Level (Note 11,12,13)	90	100	110	IRE

11. Black and white levels are referenced to the blanking level.
12. NTSC convention standardized by the IRE (1 IRE = 7.14 mV).
13. DAC ref = 3.74 kΩ, load = 37.5 Ω

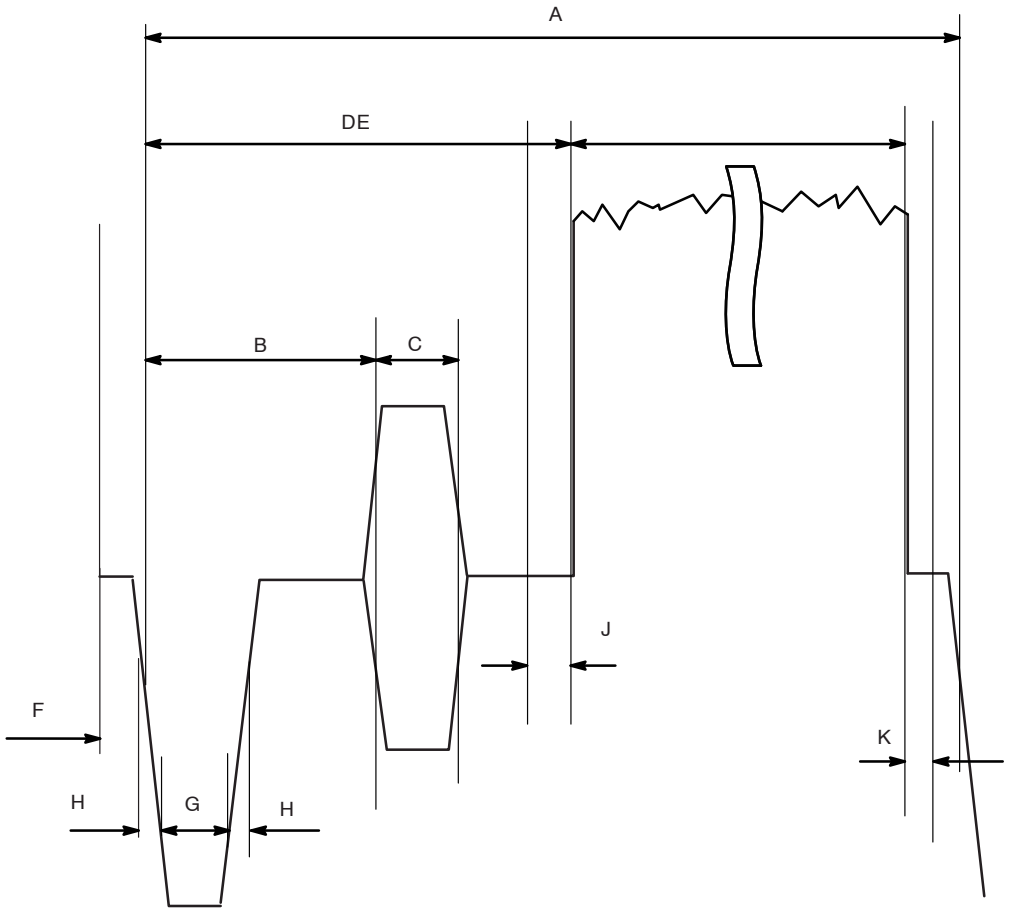


Figure 42. Video Timing

Table 43. VIDEO TIMING: SPECIFICATION FROM REC. ITU−R BT.470

	Signal	NTSC 27 MHz	PAL 27 MHz	Units
A	H Period	63.556	64.00	μs
B	Hsync to burst	4.71 to 5.71	5.60±0.10	μs
C	burst	2.23 to 3.11	2.25±0.23	μs
D	Hsync to Signal	9.20 to 10.30	10.20±0.30	μs
E	Video Signal	2.655±0.20	52+0/−0.3	μs
F	Front	1.27 to 2.22	1.5+0.3/−0.0	μs
G	Hsync Period	4.70±0.10	4.70±0.20	μs
H	Sync rising / falling edge	≤0.25	0.20±0.10	μs

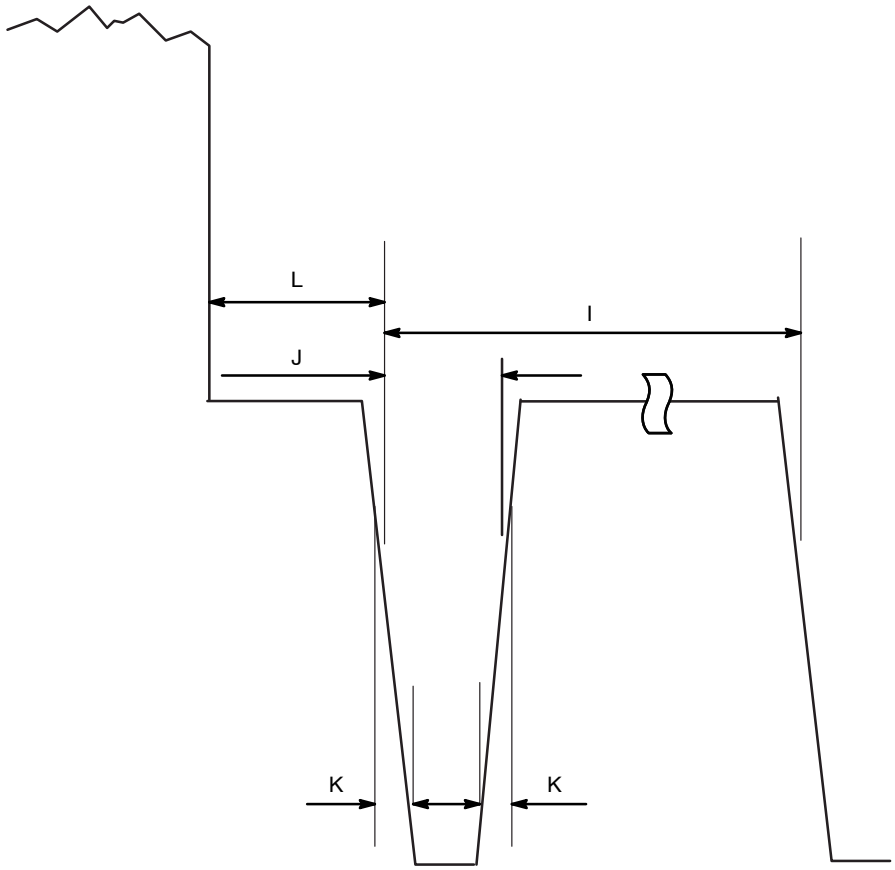


Figure 43. Equalizing Pulse

Table 44. EQUALIZING PULSE: SPECIFICATION FROM REC. ITU−R BT.470

	Signal	NTSC 27 MHz	PAL 27 MHz	Units
I	H/2 Period	31.778	32.00	μs
J	Pulse width	2.30±0.10	2.35±0.10	μs
K	Pulse rising / falling edge	≤0.25	0.25±0.05	μs
L	Signal to pulse	1.50±0.10	3.0±2.0	μs

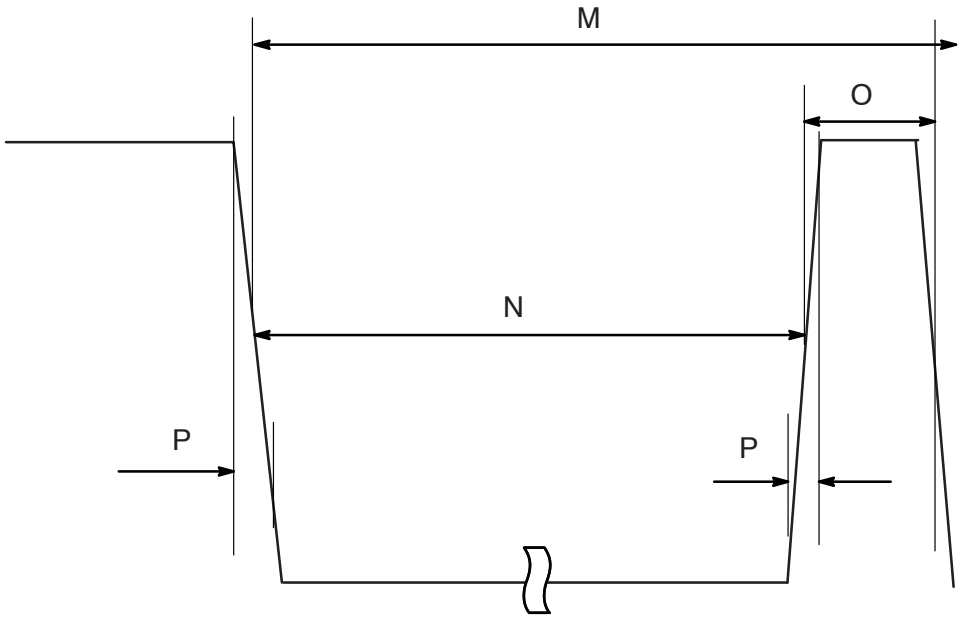


Figure 44. V Pulse

Table 45. V PULSE: SPECIFICATION FROM REC. ITU−R BT.470

	Signal	NTSC 27 MHz	PAL 27 MHz	Units
M	H/2 Period	31.778	32.00	μs
N	Pulse width	27.10 (nominal)	2.35±0.10	μs
O	V pulse interval	4.70±0.10	0.25±0.05	μs
P	Pulse rising/falling edge	≤0.25	3.0±2.0	μs

Table 46. STANDBY CURRENT CONSUMPTION

Default Setup Conditions: f_{EXTCLK}= 27 MHz, V_{DD}_REG = 1.8 V, V_{DD}IO_H not included in measurement, V_{DD}IO_S = 1.8 V, V_{DD}A_DAC = 3.3 V, V_{DD}IO_OTPM = 2.5 V, V_{DD}_PHY = 2.5 V, T_A = 70°C unless otherwise stated

Parameter	Signal	Typ	Max	Unit
Total standby current when asserting the STANDBY signal		3.2		mA
		6.9		mW
Total standby current	f_{EXTCLK} = 27 MHz	3.5		mA
		7.6		mW

Table 47. OPERATING CURRENT CONSUMPTION

Default Setup Conditions: f_{EXTCLK}= 27 MHz, V_{DD}_REG = 1.8 V; V_{DD}IO_H not included in measurement
V_{DD}IO_S = 1.8 V, V_{DD}A_DAC = 3.3 V, V_{DD}IO_OTPM = 2.5 V, V_{DD}_PHY = 2.5 V, T_A = 70°C unless otherwise stated

Symbol	Conditions	Min	Typ	Max	Unit
V_{DD}_REG		1.62	1.8	1.98	V
V_{DD}IO_H = 2.5 V		2.25	2.5	2.75	V
V_{DD}IO_H = 3.3 V		3	3.3	3.6	V
V_{DD}IO_S = 1.8 V		1.7	1.8	1.9	V
V_{DD}IO_S = 2.8 V		2.5	2.8	3.1	V
V_{DD}IO_OTPM = 2.5 V		2.25	2.5	2.75	V
V_{DD}IO_OTPM = 3.3 V		3	3.3	3.6	V
V_{DD}A_DAC		3	3.3	3.6	V
V_{DD}_PHY		2.3	2.8	3.1	V
I_{DD}_REG	NTSC HiSPi 12−bit		63.7		mA
	NTSC HiSPi 14−bit		63.6		mA
	NTSC		64.1		mA
	PAL		59.5		mA
I_{DD}IO_S	NTSC HiSPi 12−bit		3.2		mA
	NTSC HiSPi 14−bit		3.2		mA
	NTSC		3.3		mA
	PAL		3.3		mA
I_{DD}IO_OTPM	NTSC HiSPi 12−bit		0.1		mA
	NTSC HiSPi 14−bit		0.1		mA
	NTSC		0.1		mA
	PAL		0.1		mA
I_{DD}A_DAC	NTSC HiSPi 12−bit	1.2	19.54		mA
	NTSC HiSPi 14−bit	1.2	19.54		mA
	NTSC	1.2	19.54		mA
	PAL	1.2	19.54		mA
I_{DD}_PHY	NTSC HiSPi 12−bit		0.3		mA
	NTSC HiSPi 14−bit		0.3		mA
	NTSC		0.0		mA
	PAL		0.0		mA
Total power consumption	NTSC HiSPi 12−bit		185.66		mW
	NTSC HiSPi 14−bit		185.56		mW
	NTSC		185.56		mW
	PAL		177.46		mW

14. R_DAC_POS = 75 Ω, R_DAC_NEG = 37.5 Ω, R_DAC_REF= 3.74 kΩ
15. Current in single ended mode. When in differential mode the current will be 37.9 mA

Table 48. INRUSH CURRENT

Supply	Voltage	Max Current (mA)
AV_{DD}	1.8	240
$V_{DD}IO_H$	2.5/3.3	260
$V_{DD}IO_S$	1.8	15
$V_{DD}IO_S$	2.8	55
$V_{DD}A_DAC$	3.3	270
$V_{DD}IO_OTPM$	2.5/3.3	180

TWO−WIRE SERIAL REGISTER INTERFACE

The electrical characteristics of the two−wire serial register interface (SCLK, SDATA) are shown in Figure 45 and Table 49.

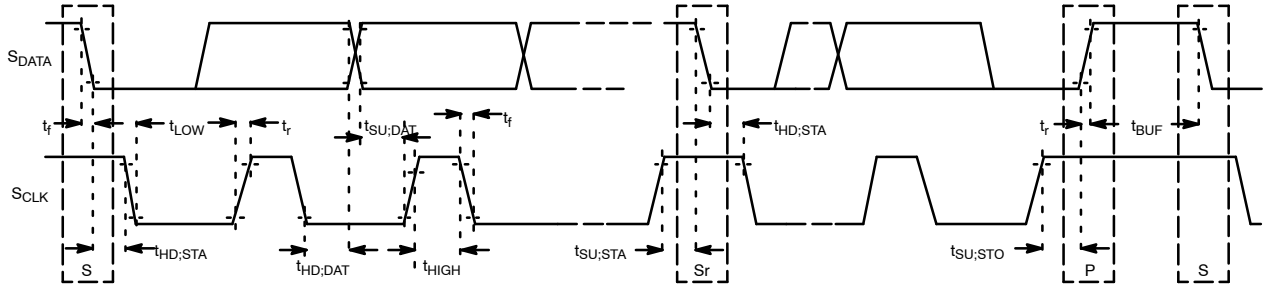


Figure 45. Slave Two Wire Serial Bus Timing Parameters (CCIS)

Table 49. SLAVE TWO−WIRE SERIAL BUS CHARACTERISTICS (CCIS)

Default Setup Conditions: f_{EXTCLK} = 27 MHz, $V_{DD}IO_H$ = V_{DD}_OTPM = 2.8 V, V_{DD}_REG = $V_{DD}IO_S$ = 1.8 V, T_A = 25°C unless otherwise stated

Parameter	Symbol	Standard−Mode		Fast−Mode		Unit
		Min	Max	Min	Max	
S_{CLK} Clock Frequency	f_{SCL}	0	100	0	400	KHz
Hold time (repeated) START condition						
After this period, the first clock pulse is generated	$t_{HD;STA}$	4.0	−	0.6	−	μs
LOW period of the S_{CLK} clock	t_{LOW}	4.7	−	1.3	−	μs
HIGH period of the S_{CLK} clock	t_{HIGH}	4.0	−	0.6	−	μs
Set−up time for a repeated START condition	$t_{SU;STA}$	4.7	−	0.6	−	μs
Data hold time	$t_{HD;DAT}$	0 (Note 17)	3.45 (Note 18)	0	0.9 (Note 18)	μs
Data set−up time	$t_{SU;DAT}$	250	−	100	−	ns
Rise time of both S_{DATA} and S_{CLK} signals (10−90%)	t_r	−	1000	20+0.1Cb (Note 19)	300	ns
Fall time of both S_{DATA} and S_{CLK} signals (10−90%)	t_f	−	300	20+0.1Cb (Note 19)	300	ns
Set−up time for STOP condition	$t_{SU;STO}$	4.0	−	0.6	−	μs
Bus free time between a STOP and START condition	t_{BUF}	4.7	−	1.3	−	μs
Capacitive load for each bus line	Cb	−	400	−	400	pF
Serial interface input pin capacitance	$C_{IN\ SI}$	−	3.3	−	3.3	pF

Table 49. SLAVE TWO−WIRE SERIAL BUS CHARACTERISTICS (CCIS) (continued)
Default Setup Conditions: f_{EXTCLK} = 27 MHz, $V_{DD}IO_H = V_{DD}_OTPM$ = 2.8 V, $V_{DD}_REG = V_{DD}IO_S$ = 1.8 V, T_A = 25°C unless otherwise stated

Parameter	Symbol	Standard−Mode		Fast−Mode		Unit
		Min	Max	Min	Max	
S_{DATA} max load capacitance	$C_{LOAD\ SD}$	−	30	−	30	pF
S_{DATA} pull−up resistor	R_{SD}	1.5	4.7	1.5	4.7	KΩ

16. All values referred to VIHmin = 0.9 V_{DD} and VILmax = 0.1 V_{DD} levels. EXCLK = 27 MHz.
17. A device must internally provide a hold time of at least 300 ns for the S_{DATA} signal to bridge the undefined region of the falling edge of S_{CLK}.
18. The maximum $t_{HD;DAT}$ has only to be met if the device does not stretch the LOW period (t_{LOW}) of the S_{CLK} signal.
19. Cb = total capacitance of one bus line in pF.

The electrical characteristics of the two−wire serial register interface (SCLK, SDATA) are shown in Figure 46 and Table 50.

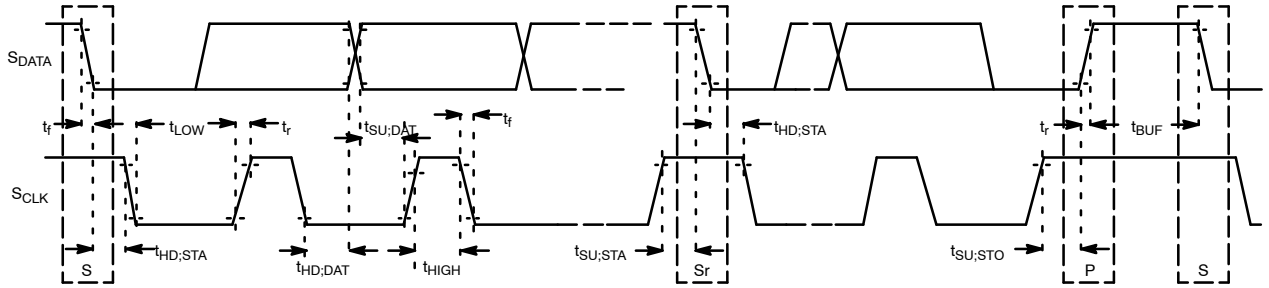


Figure 46. Master Two Wire Serial Bus Timing Parameters (CCIS)

Table 50. MASTER TWO−WIRE SERIAL BUS CHARACTERISTICS (CCIM)
Default Setup Conditions: f_{EXTCLK} = 27 MHz, $V_{DD}IO_H = V_{DD}_OTPM$ = 2.8 V, $V_{DD}_REG = V_{DD}IO_S$ = 1.8 V, $V_{DD}A_DAC$ = 3.3 V, V_{DD}_DAC = 1.2 V, T_A = 25°C unless otherwise stated

Parameter	Symbol	Standard−Mode		Fast−Mode		Unit
		Min	Max	Min	Max	
M_S_{CLK} Clock Frequency	f_{SCL}	0	100	0	400	KHz
Hold time (repeated) START condition						
After this period, the first clock pulse is generated	$t_{HD;STA}$	4.0	−	0.6	−	μs
LOW period of the M_S_{CLK} clock	t_{LOW}	4.7	−	1.2	−	μs
HIGH period of the M_S_{CLK} clock	t_{HIGH}	4.0	−	0.6	−	μs
Set−up time for a repeated START condition	$t_{SU;STA}$	4.7	−	0.6	−	μs
Data hold time	$t_{HD;DAT}$	0 (Note 21)	3.45 (Note 22)	0	0.9 (Note 22)	μs
Data set−up time	$t_{SU;DAT}$	250	−	100	−	ns
Rise time of both M_S_{DATA} and M_S_{CLK} signals (10−90%)	t_r	−	1000	20+0.1Cb (Note 23)	300	ns
Fall time of both M_S_{DATA} and M_S_{CLK} signals (10−90%)	t_f	−	300	20+0.1Cb (Note 23)	300	ns
Set−up time for STOP condition	$t_{SU;STO}$	4.0	−	0.6	−	μs
Bus free time between a STOP and START condition	t_{BUF}	4.7	−	1.3	−	μs
Capacitive load for each bus line	Cb	−	400	−	400	pF
Serial interface input pin capacitance	C_{IN_SI}	−	3.3	−	3.3	pF

Table 50. MASTER TWO−WIRE SERIAL BUS CHARACTERISTICS (CCIM) (continued)
Default Setup Conditions: f_{EXTCLK} = 27 MHz, $V_{DD}IO_H = V_{DD}_OTPM$ = 2.8 V, $V_{DD}_REG = V_{DD}IO_S$ = 1.8 V, $V_{DD}A_DAC$ = 3.3 V, V_{DD}_DAC = 1.2 V, T_A = 25°C unless otherwise stated

Parameter	Symbol	Standard−Mode		Fast−Mode		Unit
		Min	Max	Min	Max	
M_S_{DATA} max load capacitance	C_{LOAD_SD}	−	30	−	30	pF
M_S_{DATA} pull−up resistor	R_{SD}	1.5	4.7	1.5	4.7	KΩ

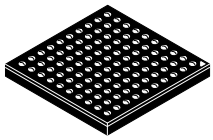
20. All values referred to VIHmin = 0.9 V_{DD} and VILmax = 0.1 V_{DD} levels. EXCLK = 27 MHz.
21. A device must internally provide a hold time of at least 300 ns for the M_S_{DATA} signal to bridge the undefined region of the falling edge of M_S_{CLK}.
22. The maximum $t_{HD;DAT}$ has only to be met if the device does not stretch the LOW period (t_{LOW}) of the M_S_{CLK} signal.
23. Cb = total capacitance of one bus line in pF.

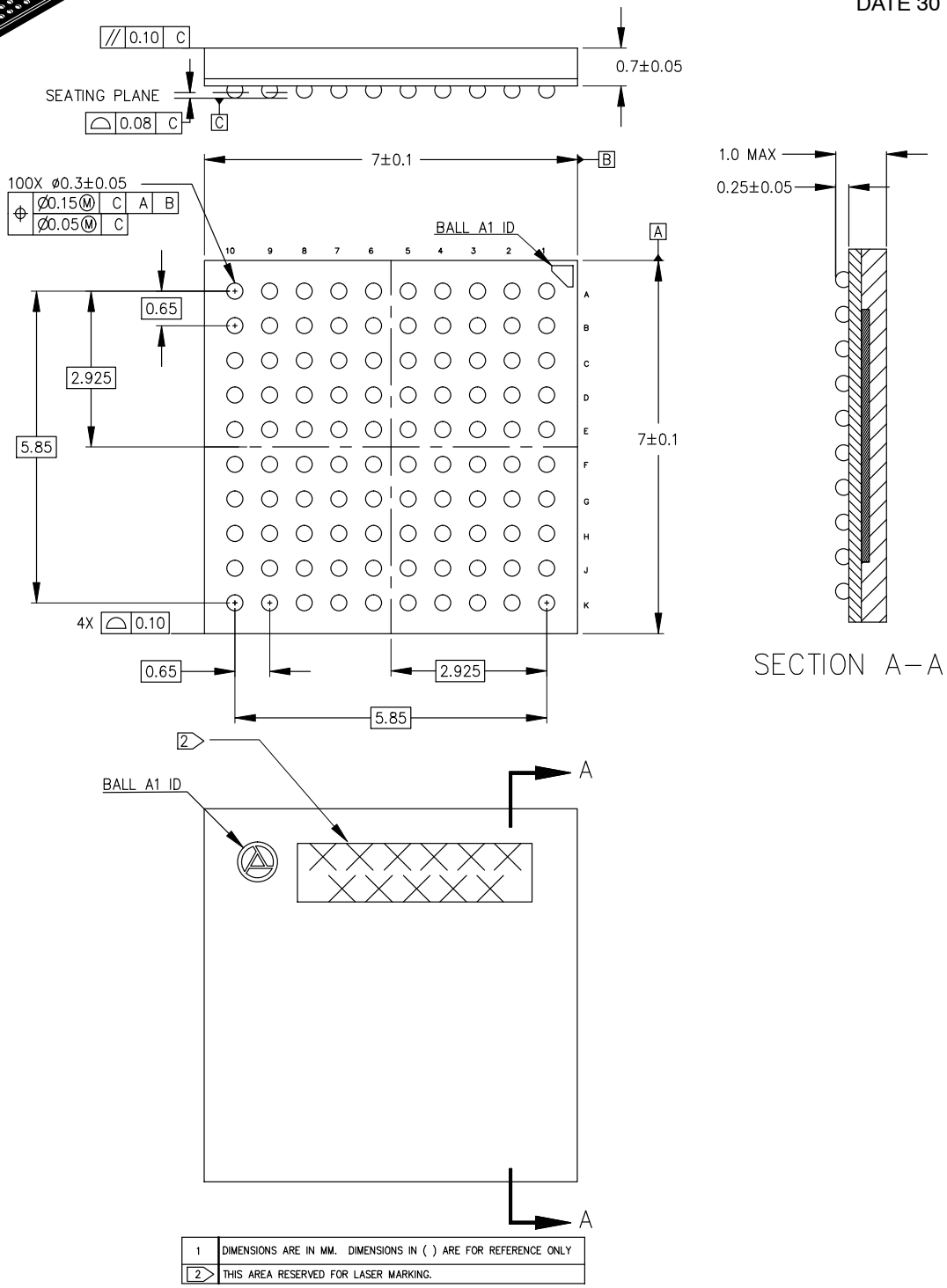
MECHANICAL CASE OUTLINE
PACKAGE DIMENSIONS

ON Semiconductor®

VFBGA100 7x7
CASE 138AH
ISSUE O

DATE 30 DEC 2014





1	DIMENSIONS ARE IN MM. DIMENSIONS IN ( ) ARE FOR REFERENCE ONLY
2	THIS AREA RESERVED FOR LASER MARKING.

DOCUMENT NUMBER:	**98AON93700F**	Electronic versions are uncontrolled except when accessed directly from the Document Repository. Printed versions are uncontrolled except when stamped "CONTROLLED COPY" in red.
DESCRIPTION:	**VFBGA100 7X7**	**PAGE 1 OF 1**

ON Semiconductor and ON are trademarks of Semiconductor Components Industries, LLC dba ON Semiconductor or its subsidiaries in the United States and/or other countries. ON Semiconductor reserves the right to make changes without further notice to any products herein. ON Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does ON Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation special, consequential or incidental damages. ON Semiconductor does not convey any license under its patent rights nor the rights of others.

© Semiconductor Components Industries, LLC, 2019

onsemi, ONSEMI , and other names, marks, and brands are registered and/or common law trademarks of Semiconductor Components Industries, LLC dba "**onsemi**" or its affiliates and/or subsidiaries in the United States and/or other countries. **onsemi** owns the rights to a number of patents, trademarks, copyrights, trade secrets, and other intellectual property. A listing of **onsemi**'s product/patent coverage may be accessed at www.onsemi.com/site/pdf/Patent−Marking.pdf. **onsemi** reserves the right to make changes at any time to any products or information herein, without notice. The information herein is provided "as−is" and **onsemi** makes no warranty, representation or guarantee regarding the accuracy of the information, product features, availability, functionality, or suitability of its products for any particular purpose, nor does **onsemi** assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation special, consequential or incidental damages. Buyer is responsible for its products and applications using **onsemi** products, including compliance with all laws, regulations and safety requirements or standards, regardless of any support or applications information provided by **onsemi**. "Typical" parameters which may be provided in **onsemi** data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. **onsemi** does not convey any license under any of its intellectual property rights nor the rights of others. **onsemi** products are not designed, intended, or authorized for use as a critical component in life support systems or any FDA Class 3 medical devices or medical devices with a same or similar classification in a foreign jurisdiction or any devices intended for implantation in the human body. Should Buyer purchase or use **onsemi** products for any such unintended or unauthorized application, Buyer shall indemnify and hold **onsemi** and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that **onsemi** was negligent regarding the design or manufacture of the part. **onsemi** is an Equal Opportunity/Affirmative Action Employer. This literature is subject to all applicable copyright laws and is not for resale in any manner.

PUBLICATION ORDERING INFORMATION

LITERATURE FULFILLMENT:
Email Requests to: orderlit@onsemi.com

onsemi Website: www.onsemi.com

TECHNICAL SUPPORT
North American Technical Support:
Voice Mail: 1 800−282−9855 Toll Free USA/Canada
Phone: 011 421 33 790 2910

Europe, Middle East and Africa Technical Support:
Phone: 00421 33 790 2910
For additional information, please contact your local Sales Representative

◊