



onsemi AP0300AT Demo System User Guide

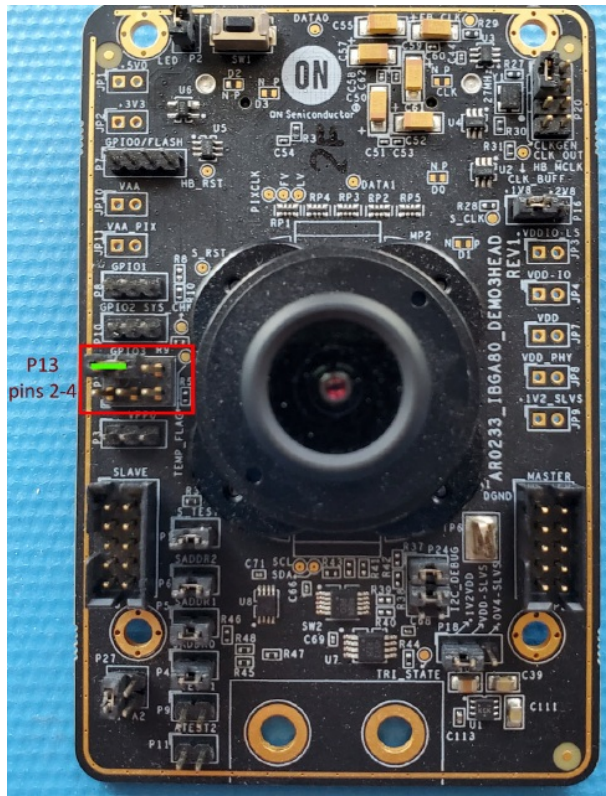
[Home](#) » [onsemi](#) » onsemi AP0300AT Demo System User Guide 

Contents

- [1 AP0300AT Demo System.](#)
- [2 Product Information](#)
- [3 Definitions and Acronyms](#)
- [4 Product Usage Instructions](#)
- [5 AP0300AT Demo System](#)
- [6 REFERENCE TABLES](#)
- [7 APPENDIX A: AP0300 AND AR0233 DEMO3 HEADBOARDS JUMPER SETUP](#)
- [8 Documents / Resources](#)
 - [8.1 References](#)
- [9 Related Posts](#)



AP0300AT Demo System.



Product Information

The AP0300AT Demo System is a combination of the AP0300 Image Signal Processor (ISP) and various demonstration and development configurations that are optimized for use with High Dynamic Range (HDR) sensors. The AP0300AT Demo System provides full auto-functions support to enhance HDR images and advanced noise reduction which enables excellent low-light performance. The AP0300 is the first ISP that offers options for user programmability.

Overview of AP0300

The AP0300 is an advanced Image Signal Processor (ISP) optimized for use with High Dynamic Range (HDR) sensors.

Required Hardware for Operation

There are three board configurations documented to support a wide variety of user setup options:

1. **Demo3 EVK (Evaluation Kit):** The EVK setup will typically consist of three essential components: the Demo3 Base Board, an AP0300 demo3 board, and a demo3 sensor headboard.
2. **MARS (Miniature Automotive Reference System):**
For a MARS setup there will be a versatile FPGA board with a USB3.0 type-B connection that provides DevWare a common interface for many onsemi imaging products. This setup allows for some more convenient mounting and testing setups.
3. **AR0233 Demo System Release:** The 2MP onsemi imaging sensor currently supported on the AP0300 ISP. Other sensors may be supported for demo-only usage now, or in the future for production usage. Includes pre-built FW binaries and supporting files required for DevWare to load the binaries.

Definitions and Acronyms

Acronym/Term	Definition
armDS / arm DS6	Arm Development Studio (6 being the current version at the time of document creation)
BSP	Boot / Security Processor, an ARM Cortex M3
FW/SW	Firmware / Software
HIP	Host Interface Processor, an ARM Cortex M3
HW Demo3 EVK	A versatile FPGA board with a USB3.0 type-B connection that provides DevWare a common interface for many onsemi imaging products.
Evaluation Kit (usually a Demo3, AP0300 demo board, and a sensor)	Hardware
MARS	Stands for Miniature Automotive Reference System, a minimized demo board in a standard form factor that more closely resemble a typical customer implementation. Also allows for some more convenient mounting and testing setups.
AR0233 Demo System Release	The 2MP onsemi imaging sensor currently supported on the AP0300 ISP. Other sensors may be supported for demo-only usage now, or in the future for production usage. Includes pre-built FW binaries and supporting files required for DevWare to load the binaries.
SDK System Release	Superset of the Demo System Release that includes example source code for writing HIP applications.

Product Usage Instructions

To use the AP0300AT Demo System, follow the below instructions:

1. Choose one of the three board configurations: Demo3 EVK, MARS or AR0233 Demo System Release. Ensure that the sensor used is compatible with the available AP0300 firmware release. Sensor compatibility is documented in the firmware packages release notes.
2. For Demo3 EVK setup, connect the Demo3 Base Board, an AP0300 demo3 board, and a demo3 sensor headboard. For MARS setup, connect a versatile FPGA board with a USB3.0 type-B connection that provides DevWare a common interface for many onsemi imaging products. For AR0233 Demo System Release setup, use the pre-built FW binaries and supporting files required for DevWare to load the binaries.
3. Load the firmware using the onsemi DevWare application. The firmware loading process is documented in the

user manual.

4. Setup an initial HIP SDK development environment by following the instructions provided in the user manual.
5. If you have any other configurations in mind, discuss them with the supporting Applications Engineer to confirm any limitations.

AP0300AT Demo System

AND9807/D

PURPOSE AND SCOPE

This document provides an overview of the using various demonstration and development configurations of the AP0300AT Rev2 Demo3 and MARS boards. Including board setup, firmware loading process with the onsemi DevWare application, and initial HIP SDK development environment setup.

AP0300 OVERVIEW

The AP0300 is an advanced Image Signal Processor (ISP) optimized for use with High Dynamic Range (HDR) sensors. The AP0300AT provides full auto-functions support to enhance HDR images and advanced noise reduction which enables excellent low-light performance. The AP0300 is the first ISP that offers options for user programmability

REQUIRED HARDWARE FOR OPERATION

The following three board configurations are documented here to support a wide variety of user setup options. For all configurations the sensor used must be compatible with the available AP0300 firmware release. Sensor compatibility is documented in the firmware packages release notes. While other configurations are possible, they should be discussed with the supporting Applications Engineer to confirm any limitations.

1. Demo3 EVK (Evaluation Kit): The EVK setup will typically consist of three essential components: the Demo3 Base Board, an AP0300 demo3 board, and a demo3 sensor headboard.
2. MARS (Miniature Automotive Reference System): For a MARS setup there will be a compatible MARS sensor, the MARS AP0300 board, (optional) MARS SerDes pair, MARS Demo3 MIPI serial (not parallel) adapter board, and the Demo3 Base Board.
3. Customer SerDes Module: This consists of a customer module with a sensor, AP0300, and Serializer on board, connected to an onsemi MARS Deserializer, MARSDemo3 serial adapter board, and the Demo3 Base Board.

REQUIRED SOFTWARE FOR OPERATION

1. Required – onsemi DevWareX: DevWare is a Powerful centralized configuration, management, and capture software for ON imaging hardware, please ensure you have a recent version. There is an additional AP0300 installer package that is only required for advanced image tuning procedures, those are standalone DevSuite applications and not covered in this document. The latest DevWare is available from this site:
<https://aptina.atlassian.net/wiki/spaces/DEVS/pages/11501573/Software+Downloads>
2. Required – AP0300 Demo Release or SDK Release Package: The Demo System Release contains pre-compiled firmware binaries to be loaded. The SDK System Releases also contain additional example code that can be modified, compiled, and then loaded. Firmware binaries from either release type can be loaded onto AP0300 EVK, MARS, and customer board stacks.

REFERENCE TABLES

Table 1. REFERENCE DOCUMENTS

Document Number	Document Title
AP0300AT–D	AP0300 Data Sheet
AND9881/D	AP0300 Technical Reference Manual
N/A	AP0300 HIP SDK Programmer’s Guide

Table 2. DEFINITIONS AND ACRONYMS

Acronym/Term	Definition
armDS / arm DS6	Arm Development Studio (6 being the current version at the time of document creation)
BSP	Boot / Security Processor, an ARM Cortex M3
FW/SW	Firmware / Software
HIP	Host Interface Processor, an ARM Cortex M3

HW	Hardware
Demo3	A versatile FPGA board with a USB3.0 type–B connection that provides DevWare a common interface for many onsemi imaging products.
EVK	Evaluation Kit – Usually a Demo3, AP0300 demo board, and a sensor.
MARS	Stands for Miniature Automotive Reference System, a minimized demo board in a standard form factor that more closely resemble a typical customer implementation. Also allows for some more convenient mounting and testing setups.
AR0233	The 2MP onsemi imaging sensor currently supported on the AP0300 ISP. Other sensors may be supported for demo–only usage now, or in the future for production usage.
Demo System Release	Includes pre–built FW binaries and supporting files required for DevWare to load the binaries
SDK System Release	Superset of the Demo System Release that includes example source code for writing HIP applications

SOFTWARE AND HARDWARE FOR HIP SDK DEVELOPMENT

onsemi does not generally provide outside software, software licenses or hardware. One exception is FreeRTOS, that is included in the HIP SDK release with some patches.

Table 3. EXTERNAL SOFTWARE AND HARDWARE

Item Name	Description	Where to get it.
ARM Development Studio, Gold (Required, Annual License)	IDE that includes compiler and DStream debugger connectivity. Gold edition includes the safety qualified compiler with maintenance updates. Used for creating custom HIP applications. Can download and try for free for 30 days from ARM, license can be installed at any time.	https://developer.arm.com/tools-and-software/embedded/arm-development-studio
UnxUtils (required on Windows only, free)	Windows port of various Unix build tools used in building the HIP applications. See AP0300_HIP_SDK_programmers_guide document for more detailed directions.	AP0300_HIP_SDK_programmers_guide section "Obtaining UnxUtils": http://sourceforge.net/projects/unxutils And http://unxutils.sourceforge.net/UnxUpdates.zip
ARM DStream JTAG Debugger (Recommended)	Debug and trace tool for on-device testing and debugging. Integrates with ARM DS.	https://developer.arm.com/tools-and-software/embedded/debug-probes/dstream-family/dstream
USB->Serial converter (Recommended, multiple/multi-port option recommended)	Allows connection to AP0300 GPIO for reading Diagnostic Serial Output (DSO), there can be three DSOs across the various processors, so a multi-port solution is a good option, or having multiple single port devices. Having the ability to connect at least two USB-Serial converter connections available should be considered required if not obtaining a DStream unit.	Available at DigiKey/Mouser/Amazon/etc. Must support 115200 baud or greater, 8N1, 3.3v. (Note 1)
Powered USB3.0 Hub capable of >=1.2 A (Required)	Having a powered USB3 hub is required as power draw when running with SerDes on can exceed the rating of most PC USB ports, resulting in loading failures or random image freezes. Must be able to supply greater than or equal to 1.2 A (6 W) on at least one USB port.	Variety of Online/Offline Retail stores. (Example: https://www.amazon.com/Sabrent-Individual-Switches-Adapter-HB-BUP7/dp/B079GT1ZVS)

1. On using Serial-USB adapters. The AP0300 can provide up to three diagnostic serial outputs (DSOs) from 3 of the internal CortexM3 cores. These operate at a baud rate of 115200, 8n1, no parity. onsemi boards will operate at 3.3v signaling levels by default, customer modules can operate at different IO voltage levels and may require a level shifter.

AP0300 EVK DEMO3 HEADBOARD STACK SETUP

See Appendix A: AP0300 Demo Board Jumper Setup for additional details on the ISP headboard jumper settings.

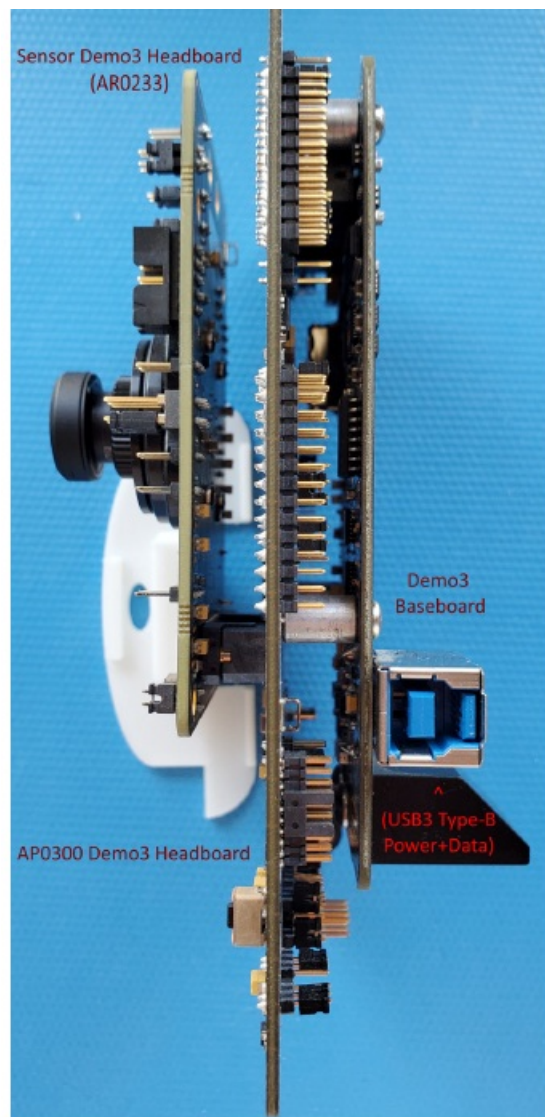


Figure 1. Demo3 Stack Side View

Figures 1 and 2 provide a side view and top-down view of a basic demo3 EVK stack, consisting of a sensor demo3 headboard (AR0233 here), AP0300 demo3 headboard, and the Demo3 baseboard.

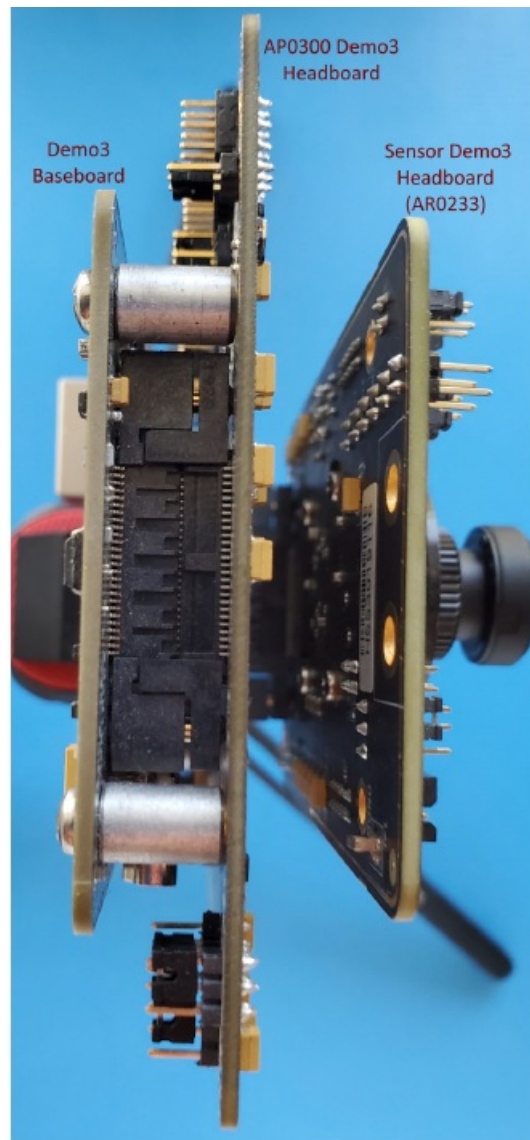


Figure 2. Demo3 Stack Top-Down View

AP0300 MARS BOARD STACK SETUP

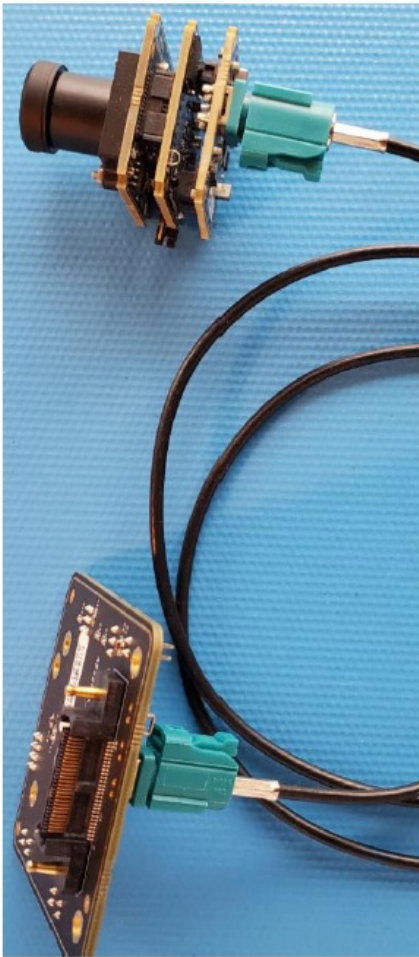


Figure 3. Assembled MARS Demo Stack

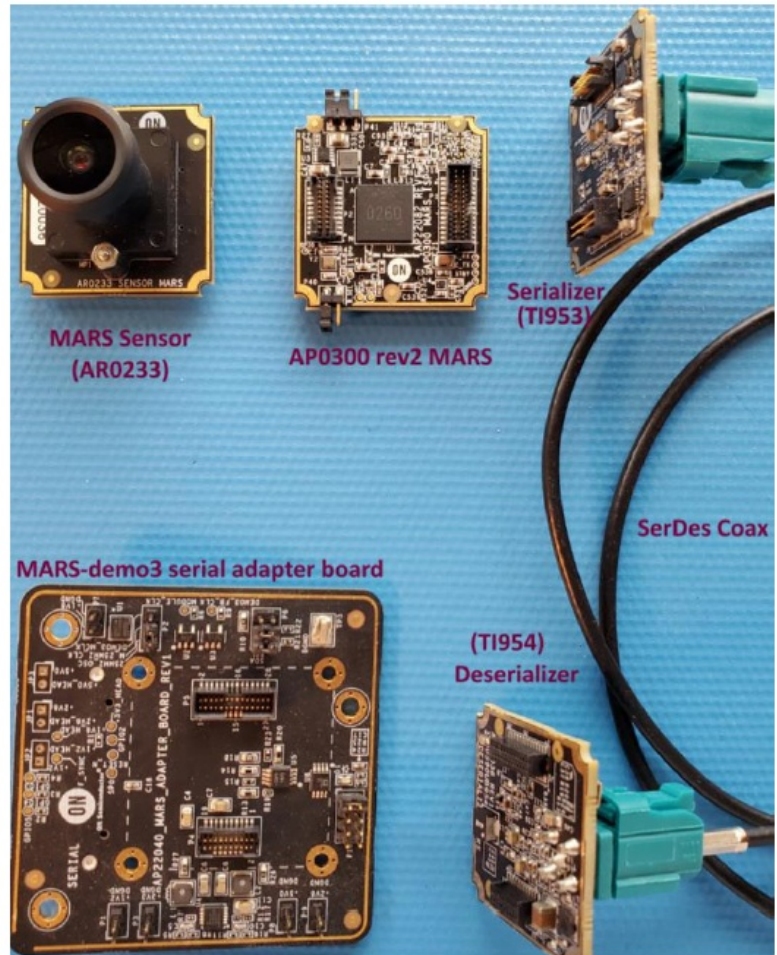


Figure 4. MARS Demo Stack Boards

LOADING FIRMWARE

The first Firmware image loaded will be a provided HIP CCA (the demo firmware), or bp_simple_streaming build that will allow streaming output viewable in DevWare.

Loading FW to RAM or Flash from DevWare

Applies to EVK and MARS board stacks, and Demo and SDK releases.

1. Assemble the EVK, MARS, or MARS+Customer board stack.
2. (ALL) Use a USB3.0 cable to plug the Demo3 board into a powered USB3.0 hub, which is plugged into a USB3.0 port on the PC that will run DevWare. Order shouldn't matter here.
 - (ALL) The Demo3 board should light up with a green LED on the side of the board facing the additional boards, indicating power is being provided.
 - (MARS, MARS+Customer) If used, a MARS SerDes pair is also likely to have a green LED indicating the pair has successfully connected.
3. (ALL) Launch the DevWare application
4. (ALL) If DevWare is configured to automatically detect devices on startup (the default configuration) then it will likely show up with the following message as seen in Figure 5. If not, then go to "File->Detect Devices to initiate the search process. Click OK when this window appears:

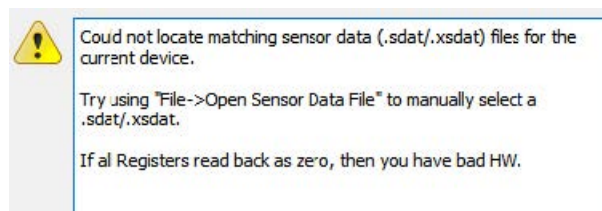


Figure 5. Load In Notice

5. If this window does not appear after clicking "File->Probe for Devices", then likely a window saying "Camera not found" appeared instead, please follow the debugging steps provided in that window to verify the board stack is properly connected, powered on, and being recognized by the PC.
6. (ALL) A window for "Startup Choices" should appear, select the "Sensor..." button to browse for a sensor data file for the device, in subsequent launches of the DevWare application you can use the "Re-Open" button and adjacent drop-down menu to select from the most recently opened sensor data files.
 - Browse to the directory where the system release file was extracted to, this should contain two directories, devware and scripts. Within the devware directory there are additional directories and supporting files, as well as the XSDAT sensor data files that will be loaded into devware.
 - Select and open the AP0300-FWUPDATE-REV2.xsd file, this may take 5-10 seconds to load as it configures DevWare.
 - A final window titled "Startup Wizard" should appear when finished, leave the checkbox "Load Initialization Settings" checked and unselect "Play on Start", then click the "Finish" button.
 - Once loaded click the "Presets" icon from the top menu bar, it is also recommended to open the "Info" and "Registers" windows/panels, "Control" will likely not be needed for this process, see Figure 6.
 - It is also suggested to enable the Python console output display with the following steps, View (from the top menu row) -> Python Console, the window can be left free or docked with other panels in DevWare.



Figure 6. Top Icon Menu Options

7. (ALL) The Presets window should now be open, this is loaded from an INI file located in the same directory, and with the same name, as the previously selected XSDAT file, and should look similar to Figure 7.

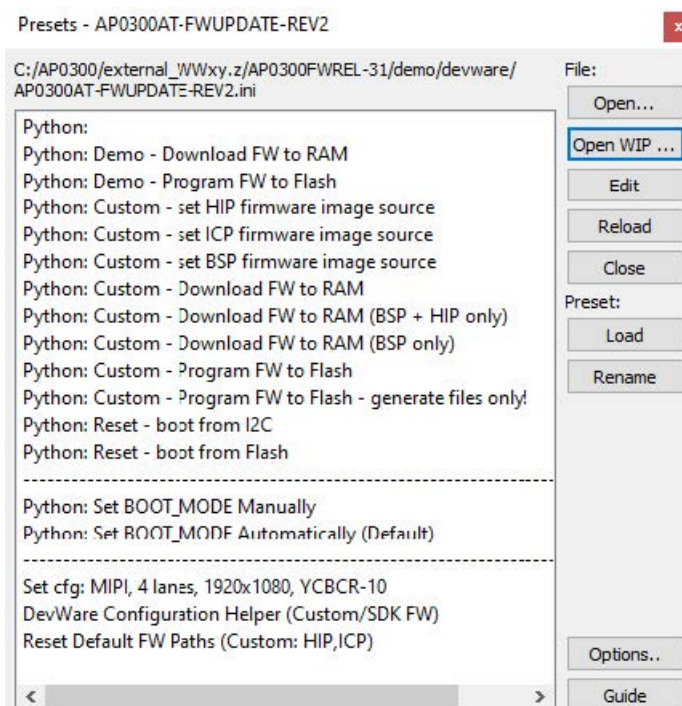


Figure 7. FWUPDATE Presets Window

8. (ALL) From the register window/tab confirm that DevWare is already able to read out a value of decimal 356 or hex 0x0164 for the AP0300 Chip ID register, address 0x0000, in the "HIP: i2cs_regmap" page, and that lower down in that register panel is a green "Comm: Working" status:
 - Note a value of decimal 300 in the Chip ID register is for Rev1 parts, which should not be used. Ensure you see a decimal value of 356 for Chip ID on Rev2 parts.

- The below figures show expected Chip ID and communication appearance in DevWare.

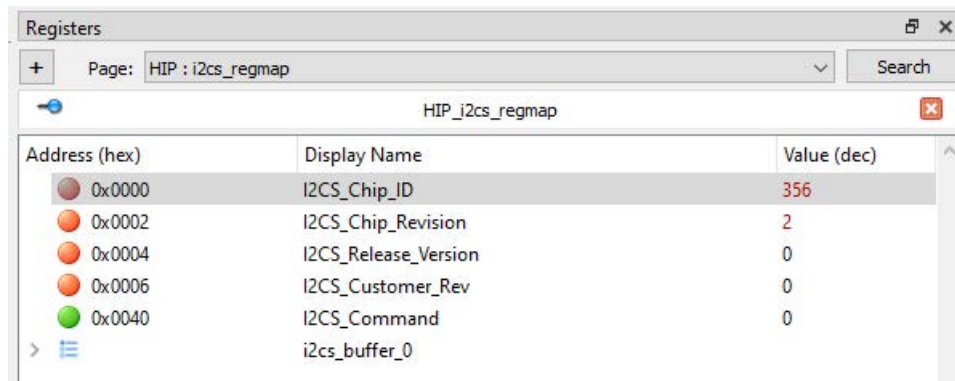


Figure 8. HIP I2C Registers Page

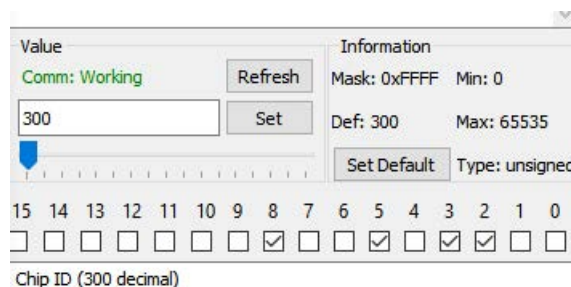


Figure 9. HIP I2C Communication Status Field

- (ALL) First debugging step if these fields are not as expected is to click the "Refresh" (with the Chip ID register selected) or "Refresh All" buttons in the lower options area.
 - (EVK) If not working then power off the system and close DevWare, check the assembly of the board stack, and try again. If you still don't see the correct Chip ID with the evaluation kit setup then please contact an FAE or AE.
 - (SerDes) With a SerDes pair in your board stack the Deserializer probably requires setup before the I2C back-channel communication will work. DevWare will often be able to do this behind the scenes automatically, but if not, this is the most likely cause of not seeing communication with the AP0300. To fix this talk to your FAE/AE about the SerDes configuration INI file. These configuration files can often be placed in the root of your DevWare install directory (often C:\Aptina Imaging) where they can be detected and automatically ran by DevWare in the future.
9. At this point the process diverges depending on which firmware you are loading. Continue with the next appropriate section:
- Demo (HIP CCA – Camera Control Application) FW, see section: Loading the Demo FW (HIP CCA) to RAM or Flash
 - HIP SDK FW, such as bp_simple_streaming, see section: Loading a HIP SDK demo FW build to RAM or flash

Loading the Demo FW (HIP CCA) to RAM or Flash

The Demo Camera Control Application provides DevWare an interface to control and configure

1. (MARS/Customer) From the presets window run the "Python: Set BOOT_MODE Manually" preset, read, and click OK on the pop-up information window.
2. (ALL) Select and run the "Python: Demo – Download FW to RAM" or "Program FW to Flash" depending on which you'd like.
3. (MARS/Customer) An information window will likely appear asking you to confirm the Boot Mode line is properly set to 1 for I2C boot. Depending on your setup this may be already done, and you can

- click OK to continue. If not, there may be a jumper to set to correctly configure the boot mode for your device.
4. (ALL) The python console, available in View
(from the top menu row) → Python Console, will have readouts on the progress of the download. Depending on the I2C speed, image size, and if SerDes is used this can take 20 seconds to several minutes.
 5. (Loading to RAM) Continue to step 7
 6. (Loading to Flash) With the Demo FW the remainder of the setup involves giving DevWare more complete control of the system and further setup steps. Now that the flash has been loaded the following steps, depending on board stack, will boot up AP0300 with the download Demo FW.
 - (EVK) Select and run the "Python: Reset – boot from Flash" preset.
 - (MARS/Customer) If jumper controlled: power off or unplug the board stack and switch the boot mode to 0=boot from flash. Then plug the device back in. Otherwise, you might have another preset to set the boot mode.
 - If a power-cycle is needed, you will need to reconnect to the Demo3 base board in this situation, select File "Detect Devices" in DevWare. Select the AP0300AT-REV2.xsdat file, jump to step 8.
 7. (ALL) Load the AP0300AT-REV2.xsdat file, use "File→Open Sensor Data File..." to open this XSDAT. Do not use the "Open..." button in the presets window, as that would not load the XSDAT file, only the INI, and result in error messages about unknown registers.
 8. (ALL) When the Startup Wizard window appears, ensure that "Play on Start" is deselected, then click "Finish" to continue.
 9. (ALL) If the presets window is not open, open it.
 - For immediate image output, run the "Demo Initialization" preset to finish configuration of the AP0300, sensor, and DevWare. This Preset should also enable streaming in DevWare. Once completed, which may take a minute, DevWare should display the image stream.
 - For more details on usage and available features with the HIP CCA firmware, see the section "Running the HIP CCA Firmware".

Loading a HIP SDK demo FW build to RAM or flash

For an example of an SDK FW demo see `bp_simple_streaming`, which will be used as an example where needed in the rest of this section. See "Description of included Demo firmware images" for information on the different pre-built FW images.

1. (MARS/Customer) From the presets window run the "Python: Set BOOT_MODE Manually" preset, read and click OK on the pop-up information window.
2. (ALL) Use the "Python: Custom – set HIP firmware image source" to open a file browser window and find the .cif or .axf compiled images that you want to load. The included system release images are in the "firmware" sub-directory at the same location as the XSDAT and INI files unpacked from the system release file. For example, the simple streaming example that may be loaded is labeled `bp_simple_streaming_hipfw.cif`.
3. (ALL) Run the "Python: Custom – Download FW to RAM" or "Program FW to Flash" preset.
4. (MARS/Customer) An information window will likely appear asking you to confirm the Boot Mode line is properly set to 1 for I2C boot. Depending on your setup this may be already done, and you can click OK to continue. If not, there may be a jumper to set to correctly configure the boot mode for your device.
5. (ALL) The python console, available in View (from the top menu row) → Python Console, will have readouts on the progress of the download. Depending on the I2C speed this can take 15 seconds to over a minute.
6. (Loading to RAM) Continue to step 8

7. (Loading to Flash) Sensor setup is contained in the FW image and is not dependent on DevWare.

To load the image from flash, follow the settings for you setup below:

- (EVK) Select and run the "Python: Reset – boot from Flash" preset.
- (MARS/Customer) Power off or unplug the board stack and switch the boot mode to 0, boot from flash. Then plug the device back in.

8. (ALL) Configure DevWare to display the stream, this depends on the sensor and image loaded. Here are some options to try:

- Use one of the presets for "Set cfg: <settings>"
- Open the "Control" window, gear icon from the top icon menu, go to Diagnostics->Serial Receiver, configure the Connection Type to MIPI, 4-Lanes, and Serial Bit Depth depending on the loaded firmware (see section below titled: HIP Firmware Output Format, DevWare Setup). Then go to Diagnostics->Sensor Advanced and configure sensor output or press the "detect" button.

9. (ALL) Press the "Play" icon if not already highlighted, DevWare should now be displaying the image stream.

GENERATE BINARY FILE(S) FOR LOADING TO RAM OR FLASH OUTSIDE OF DEVWARE

1. Follow the previous section, Loading FW to RAM or Flash from DevWare (EVK and MARS, Demo and SDK releases) for loading a HIP SDK build.
2. Select "Python: Custom – Program FW to Flash – generate files only!", or the "Download to RAM" preset.
3. The built binary file to be loaded is stored in the same firmware directory as the selected HIP image, the file will be called custom_flash_image.bin. The path including filename are printed in the DevWare Python console panel.

DESCRIPTION OF INCLUDED DEMO FIRMWARE IMAGES

The AP0300 Demo and SDK System Releases contain a variety of pre-built binary images for demonstration of system functionality and as a programming reference in the case of the SDK releases where the source code is included. These FW images can be found in the /devware/firmware/directory.

Table 4. INCLUDED FIRMWARE IMAGES

FW Image Name (.cif)	Description
bl2	2nd stage of the bootloader image, part of the firmware and supporting files loaded onto the BSP core.
icpfw	FW for the ICP core.
hipfw	HIP CCA Demo FW, Camera Control Application, allows and requires DevWare to set up sensor, loaded by the "Demo- <>" presets.
bp_simple_streaming_bypass_hipfw	Automatic sensor bring-up with output format 12 bit Bayer Linear mode, bypassing the AP0300 processing. The Rotate 180 variant flips and mirrors the output image.
bp_simple_streaming_hipfw, bp_simple_streaming_8_bit_hipfw	Automatic sensor bring-up with output format 10bits per component, 20bits per pixel, YCbCr-10. The 8-bit variant outputs at 8 bits per component, 16 bits per pixel

HIP FIRMWARE OUTPUT FORMAT, DEVWARE SETUP

The FWUPDATE INI includes a preset “DevWare Configuration Helper” that will typically be able to correctly detect the FW image loaded and perform these set up steps. Depending on the sensor used and FW loaded, DevWare will require different configuration options to correctly display image data. The below table can be used as a reference to help configure the options found in the Control (gear icon in top icon menu) Diagnostics Sensor Advanced Sensor Output. For all sensors and output the Control Diagnostics Serial Receiver settings should be configured as follows: Connection type = MIPI, Lanes = 4 lanes.

Table 5. DevWare CONFIGURATION GUIDANCE

Loaded HIP FW binary	AR0233 Default DevWare Configuration
hipfw	Automatic, depends on selected pre-set. Open AP0300AT-REV2.xsdat for option pre-sets.
bp_simple_streaming_bypass_hipfw	1920×1080, Bayer-12
bp_simple_streaming_hipfw	1920×1080, YCbCr-10
bp_simple_streaming_8_bit_hipfw	1920×1080, YCbCr-16

RUNNING THE HIP CCA FIRMWARE

The HIP Camera Control Application provides an interface that emulates additional register control sets. It is strongly recommended to load the HIP CCA firmware application to Flash memory for usage, as it otherwise will take some time to load and provides an easy way to reset settings with a reboot.

The onsemi suggested settings can be run with the “Demo Initialization” pre-set. This results in all required steps for the ISP and DevWare being completed for display in the DevWare viewing window. There is also a 20-bit Bayer bypass operating mode, which is a direct feed from the sensor output prior to any ISP functionality.

1. Operating Mode and General Control Presets:

- Demo Initialization – The onsemi suggested optimal demo settings. This performs all required setup steps for the ISP and DevWare viewing with no additional user input required.
- Clean Initialization 20-bit Bayer Bypass –direct output feed from the image sensor, bypassing all ISP functionality. The clean initialization means that any existing settings will be cleared and reset to default values.
- Enter Linear 12-bit Bayer Calibration mode – A linear output mode used only for some existing DevSuite tuning tools.

2. Output Control Presets:

- Output IPIPE – Default output path from the ISP imaging pipe, and demo initialization. Typically, YCbCr formatted.
- Output RX Bypass – Bypass all ISP imaging pipeline and output the image data + format from the sensor, Bayer format.
- There are some additional sensor-specific RX Bypass configuration presets due to various sensor output limitations, use the one appropriate for the sensor in use.
- Output Extract0 [multiple presets] – Output from the configurable extraction points within the ISP imaging pipe. May be used in some specific tuning but typically not required.

3. Sensor Specific Presets: Various sensor-specific setups. Note only AR0233 is supported for production use,

any other sensors listed are currently limited demo usage only.

4. Flash Configuration Controls: See section HIP CCA Flash Configuration Usage below.
5. Get_icp_sw_error_info: If the HIP CCA stops responding or outputting image data, this preset may provide detailed error messaging related to the cause of the error.

Many settings can be modified during camera operation using the DevWare register control interface. For more details on the HIP CCA registers related to tuning image quality, see the AP0300 Customer Tuning Guide. When making a change to register value(s), the “Start–Stream” preset or top menu icon should be run in order for changes to take effect.

HIP CCA FLASH CONFIGURATION USAGE

When the HIP CCA comes out of reset, it can be configured to automatically load a saved sequence of initialization settings. This feature allows the camera module to immediately start streaming image data on power up and is useful for testing settings across lab setups or on a vehicle. The control interface for the flash configuration feature is available via the “Flash Configuration Controls” pre–set group in the AP0300AT–REV2 INI file.

To provide the filename and a flash configuration file index, enable the User Control Panel DevWare window under View “User Control Panel”. Note that that file index value is 1–indexed, a zero is not a valid index value.

Table 6. HIP CCA FLASH CONFIGURATION PRESET OPTIONS

Pre–Set Name	Description
Flash Recording Mode (On/Off)	<p>Toggles recording of settings changes to the selected flash configuration slot on the camera. While ON, any settings changes that take place will be saved, until the recording mode is turned off.</p> <p>The filename and file index must be set in the User Control Panel prior to executing this pre– set.</p>
List Stored Flash Configuration Files on Camera	Show the previously saved recordings on the camera, by filename, in index order.
Save Specified Flash Config file from Camera to PC (text + bin)	Download a binary and text version of the saved flashed configuration, as specified by the Config File Index value, to the DevWare computer from the camera module.
Format and Initialize the User Flash Partition	Initialize the flash memory on the camera for the flash config usage. This will often need to be done prior to recording to flash the first time on a new headboard/camera. Formatting clears out any saved flash configurations on the camera, save any desired configurations prior to a reformat.
Set specified Flash Configuration file to be used for future bootup–to– Flash	Tells the HIP CCA to attempt to load the saved flash config on boot, as specified in Config File Index when this pre–set is executed. If valid, the saved sequence of settings will be loaded into the application and the module can start streaming immediately on boot.
Clear Specified Flash Configuration	Wipe the saved flash configuration from the camera module from the slot specified in Config File Index.
Load Flash Configuration .bin file from PC to Camera	Upload a previously saved flash configuration binary from the DevWare computer to the camera module.

BOOT SEQUENCE TECHNICAL INFORMATION

This section provides some technical background on the process and protocol used to load firmware onto the AP0300, either to RAM or attached Flash chip. These procedures and protocols are used by the INI and Python

files by DevWare. Customers are encouraged to implement their own scripts for their own specific development and production environments if needed. The included DevWare scripts are a good reference for any separate implementation.

PROTOCOL OVERVIEW

The protocol for loading or programming firmware onto the AP0300 RAM or flash is a straight-forward two-stage process. In both stages, the host must ensure that the BOOT_MODE pin is held high while toggling the RESET pin to put the AP0300 into I2C boot mode. It then sends a series of commands over I2C to deliver a binary image to the AP0300. The table below describes the I2C commands used in this process.

Table 7. I2C BOOT COMMANDS

Command Name	Command Code	Parameters	Notes
START_NORMAL_LOAD	0x0000	(none)	Begin loading binary data. Used to initiate either the first-stage boot or a second-stage boot in which only the HIP and ICP firmware images are loaded.
LOAD_CHUNK	0x0001	(1 – 256 bytes of data)	All but the last block contains the next 256 bytes of data. The last block contains the remaining data bytes.
END_LOAD	0x0002	(none)	End the loading of binary data.
START_PROGRAMMING	0x0010	(none)	Begin programming the flash.
PROGRAM_CHUNK	0x0011	(256 bytes of data)	Each block of 256 bytes of data will be written sequentially into the flash, starting at sector 0.
END_PROGRAMMING	0x0012	(none)	End the programming of the flash.

There are four command sequences that are supported:

1. The first stage boot always consists of a START_NORMAL_LOAD command, followed by one or more LOAD_CHUNK commands, and ends with an END_LOAD command. This stage is the same regardless of whether a load-to-RAM or program-to-flash operation is being performed. Only one of the following steps will be taken following this first stage.
 - a. The second stage boot when loading only HIP and ICP firmware images into RAM is the same as the first stage boot: a START_NORMAL_LOAD command, one or more LOAD_CHUNK commands, and an END_LOAD command.
 - b. The second stage boot when programming the flash starts with a START_PROGRAMMING command, followed by one or more PROGRAM_CHUNK commands, and ends with an END_PROGRAMMING command. Note that in each command sequence the length of the binary data being transferred is not explicitly sent. In the case of a load-to-RAM operation, the AP0300 calculates the total length of data it expects to receive from fields embedded at various places in the binary data. In the case of a program-to-flash operation, the AP0300 programs as many 256-byte chunks to the flash as it receives.Each command sent returns an error code to indicate if the command successfully completed or not. The host must wait for a response from the AP0300 before attempting to send the next command. If any command

returns a non-zero error code, then an error condition was encountered, and the entire operation must be terminated.

SENDING COMMANDS TO THE AP0300 OVER I2C

The AP0300 I2C slave interface is register-based with 16-bit register addresses and 16-bit wide registers. It is big-endian, meaning that the first byte of a 16-bit value is the most significant byte and the second byte is the least significant byte.

In previous onsemi ISPs and SoCs there were hundreds of registers and firmware variables accessible via I2C, but on the AP0300 there are only four read-only ID registers, a read-write COMMAND register, and a 256-byte read-write parameter pool. This trimmed-down register set supports a command-based interface only. When running the HIP CCA “demo” firmware, additional registers are emulated to provide a register interface like what is found in onsemi sensors and previous ISP/SoC products.

Each command is sent to the AP0300 using the following procedure:

1. The host reads the COMMAND register (0x0040) and checks bit 15 to make sure it is zero. This indicates that the AP0300 is ready to receive a command.
2. The host writes any parameter data into the 256-byte command buffer (0x0100). It can do this in a single I2C transaction or multiple transactions as desired. The parameter pool is essentially a 256-byte RAM, so no particular order of writes must be performed.
3. Once the parameter pool has been written, the host writes the command value to the COMMAND register (0x0040) with bit 15 set to 1.. For example, to send the START_NORMAL_LOAD command, the host would write 0x8000 to the COMMAND register. This signals the AP0300 firmware to service the command.
4. The host polls the COMMAND register (0x0040), waiting for bit 15 to go low. This signals the host that the AP0300 has completed processing the command.
5. The host extracts the command error code from the lower 15 bits of the COMMAND register. If this value is non-zero, then the command failed, and the code indicates the reason for the failure. Otherwise, the command succeeded. (For boot commands, the AP0300 only returns error codes and does not alter the contents of the parameter pool.)

SPECIAL HANDLING ON THE LAST COMMAND OF THE LOAD-TO-RAM BOOT PROCEDURE

When the AP0300 second-stage boot code (BL2) receives the END_LOAD command, after loading the HIP firmware image, it leaves bit 15 of the COMMAND register set. This is the same state that the COMMAND register is in after power on or reset, so HIP firmware that implements its own I2C command handler can indicate that it is ready to receive I2C commands by clearing the COMMAND register regardless of whether it was loaded from flash or over I2C. This means that when the host sends the END_LOAD command at the end of the second-stage boot, it should not wait for bit 15 of the COMMAND register to go to 0.

SERDES LINK INITIALIZATION

When a serializer/deserializer chipset pair (aka “SerDes”) is used between a host I2C master and the AP0300’s I2CS interface, it must first be properly set up before the AP0300 can be accessed. Only the TI 953/954 SerDes pair has been used with the AP0300 at this point, but others should work as well.

The first step is the deserializer initialization, setting up the I2C backchannel is required before the serializer and then the AP0300 and other devices on the same I2C bus can be interacted with. Once initialized the serializer can be configured, then the AP0300 and associated devices can be interacted with.

Detailed implementation of the TI953/TI954 SerDes initialization can be found in the DevWare installation directory, by default “C:\Aptina Imaging\apps_data\Serdes\Tl\Ti.ini”. Other initializations can also be found in that SerDes directory.

ARM DEVELOPER STUDIO SETUP:

Please start by following the PDF located in the HIP SDK "doc" folder

AP0300_HIP_SDK_programmers_guide.pdf. , Arm Developer Studio can be downloaded directly from ARM after signing up for an account and includes a 30-day trial, license keys can be entered at installation or during the trial. ARM DS Gold is the recommended version for the safety qualified compiler and maintenance updates. Extract the SDK release where it is easy to navigate to and without any spaces in the path name.

Once setting up your first project select "New Project" from the Project Explorer view, or "File → New → Project". Select the "Makefile Project with Existing Code" from the C/C++ folder in the new project wizard and click "Next". Select ARM Compiler 6. Then select "Browse" in the "Existing Code Location" section, browse to the HIP SDK firmware location on your computer (contains the code for various example/starter projects, may need to be extracted). From this point there are two different options for which file level you select.

Option 1. (Preferred) Select the root HIP SDK folder, containing the src, contrib, and other directories, this requires a little more setup initially but allows you to build all products from the same project individually and provides the IDE more context for code parsing. Then for each demo application you can right-click and create two build targets, name one "all" and one "clean", without the quotation marks. Now you can double click all or clean to build or clean the demo application. This setup also allows the IDE to automatically have most or all the function definitions available, such as for tracing where functions are defined or used, and auto-completion.

Option 2. Select the bp_<some demo variant> folder from the HIP SDK folder. This option requires less initial setup to get building, but more setup later if you want proper context highlighting. Once that folder is selected, click "OK" and it should automatically populate the top "Project Name" field, which can be further edited at this point if desired, then click "Finish".

ARM DS will then spend ~10–60 seconds setting up some more things, you can tell those items to run in the background.

That should be all that is required to setup ARM DS to build the HIP FW images that can be loaded via DevWare. You can run the build process using the top menu bar "Project→Build All" (Ctrl+B), once the build completes, if there were no errors, the last couple lines of the console output should be about the full image and header CRC values.

ARM DS TIPS

- Could not make or find output/debug directory, or unable to make clean because command not found: There is likely an issue with the UnxUtils environment setup, either the UNIX_UTILS_ROOT environment variable, which should point to the usr\local\wbin directory where you installed UnxUtils, or the order in the PATH environment variable is preceded by another application of the same name (Cygwin could be related). See the HIP SDK programmers guide for complete instructions on the UnxUtils setup.

OTHER DEBUGGING TIPS:

- Unable to load because last command not clear on initial power-on, I2CS_Command register always 0x8000 – This is usually due to the part being set to boot from flash, you may be able to call the boot from i2c preset, or you may need to swap/remove a jumper to fix. See Appendix A: AP0300 Demo Board Jumper Setup for jumper setup help.
- Register Read error messages in DevWare: There's several possible reasons for this:
 - Confirm in the DevWare "Registers" panel values section that there's green text saying "Comm: Working".
 - Confirm that the correct XSDAT for your INI file is loaded via the File "Open Sensor Data" file menu, in most cases you will start with loading the AP0300AT-FWUPDATE-REV2.xsdat file, only once the Demo firmware (HIP CCA) is running should the AP0300AT-REV2.xsdat be loaded.
 - Don't load the AP0300AT-FWUPDATE-REV2.ini or AP0300AT-REV2.ini file through an existing presets window. This will skip loading the register definitions from the associated XSDAT files.
- Other streaming errors/failures to stream: Ensure that the version of ICP FW loaded is from the same release

as the loaded HIP FW image, or the same HIP SDK release version for custom images.

- Builds from clean start, but partial rebuilding fails, or does not build at all, and error seems to be a partial path, such as “C:\Program”. Make does not support spaces in paths, if your project is in a path with a space in it you will likely have problems compiling. Alternatively, ARM defaults its installation location to within “C:\Program Files[...]", by default the standard library includes should not be part of the dependency files used for partial rebuilds, but if a compiler flag gets removed from the HIP SDK common/build_env.mk or build_rules.mk setup files, then that may also cause this sort of error.

APPENDIX A: AP0300 AND AR0233 DEMO3 HEADBOARDS JUMPER SETUP

The following image shows a functional jumper configuration for the AP0300 Demo3 headboard Rev1, which contains an AP0300 Rev2 part. Note that PCB rev is not the chip revision. See the AP0300_VFBGA169_DEMO3HEAD_REV1 schematic for more details and default jumper configuration and functionality, see below the image for key jumpers to be aware of and check.

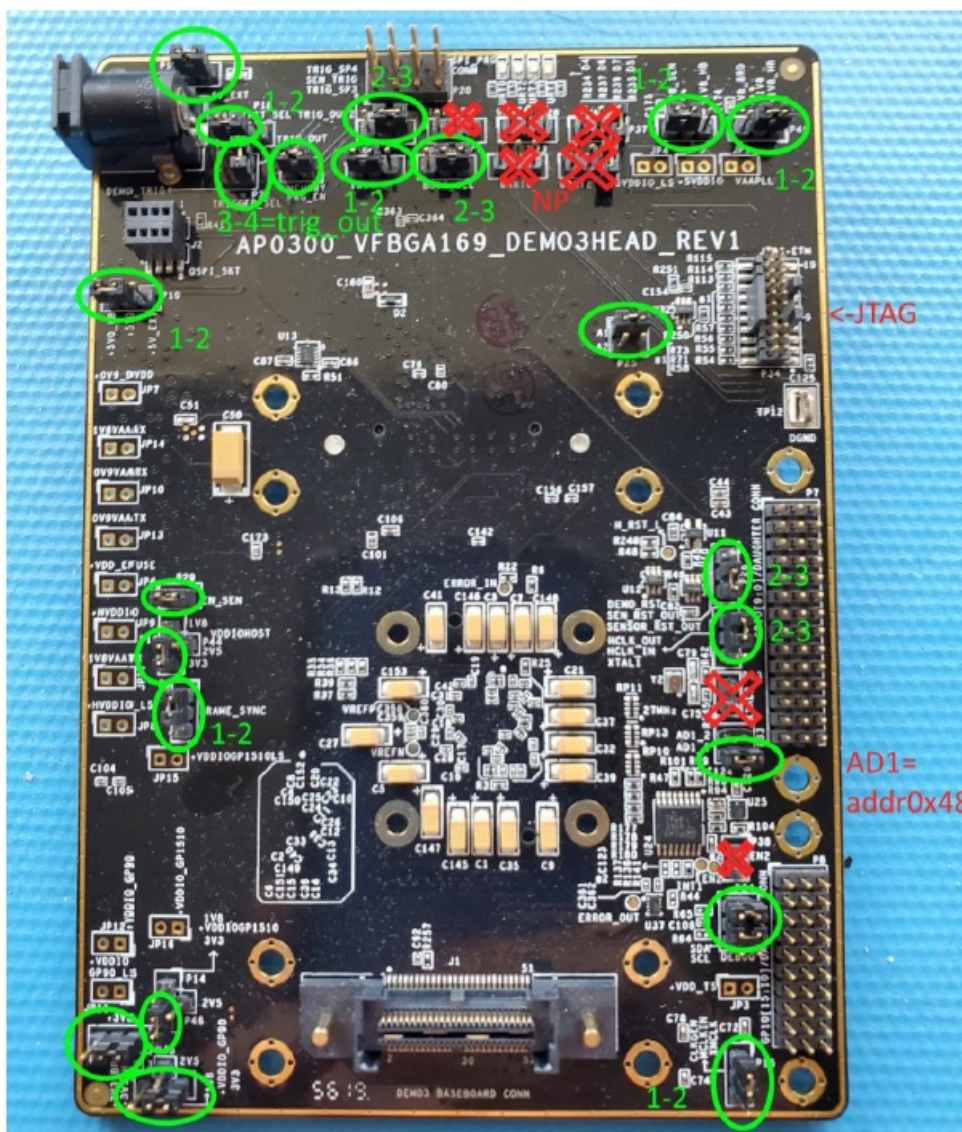


Figure 10.

- For some recent headboards, the jumper on P33 will likely need to be changed for compatibility with the INI files. Those expect the GPIO expander, which provides programatic control of the BOOT_ MODE and RESET_n pins, to be at address 0x48. Unfortunately the default here was changed for one batch of built headboards.

- Optionally, the 4 UART–TX diagnostic serial output (DSO) pins may have any jumpers removed, (P23, P28, P30, and P37). If you are connecting standard headers to a USB–UART converter these are typically easier to connect to than the P8 header groups, which have a less common pin pitch and size.
- If you want to use a trigger signal from the demo3 baseboard, check the configuration of jumpers P22 and P35. Those will typically need to be changed to allow for passing through the trigger signal.
- Other than these noted items to be aware of, the default jumper configuration is typically correct for operation powered by the demo3 baseboard.



Figure 11.

- If using AR0233 trigger operating modes from the AP0300 or demo3 baseboard the jumper on P13 of the sensor headboard should be changed as shown in the image above, to pins 2–4. This allows the trigger output signal from the AP0300 demo3 headboard to connect to the sensor’s GPIO3 pin.

Arm, Cortex, and the Arm logo are registered trademarks of Arm Limited (or its subsidiaries) in the EU and/or elsewhere. DSTREAM is a trademark of Koninklijke Philips N.V.

Python is a trademark of the Python Software Foundation.

Windows is a registered trademarks of Microsoft Corporation.

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

Documents / Resources

	<p>onsemi AP0300AT Demo System [pdf] User Guide Demo3 EVK, AP0300 demo3 board, MARS, AP0300AT Demo System, AP0300AT, Demo System</p>
---	--

References

-  [UnxUtils download | SourceForge.net](#)
-  [Intelligent Power and Sensing Technologies | onsemi](#)
-  [Intelligent Power and Sensing Technologies | onsemi](#)
-  [Confluence](#)
-  [Amazon.com. Spend less. Smile more.](#)

