
How to configure VD55G1, VD25G4, VD55G4 and VD65G4 in auto wake-up mode

Introduction

The VD55G1, VD25G4, VD55G4, and VD65G4 have an auto wake up (AWU) feature. This feature allows the sensor to monitor a scene in low-power mode. It triggers the host when it detects a significant change. In this document, unless otherwise stated, the sensor refers to these devices.

In AWU mode, the sensor does not output an image stream by default. Instead, the internal firmware monitors one or more selected regions of interest and evaluates scene changes using an embedded detection mechanism. When the programmed criteria are met, the sensor asserts a wake-up event through a GPIO interrupt.

After the wake-up event, the sensor can either transition directly to normal streaming mode or return to SW_STANDBY and wait for additional host instructions. This flexibility allows the system to accommodate the host wake-up latency and the image reception pipeline requirements, whether the image data is received over MIPI, SPI, or I3C.

This application note explains:

- How the AWU feature works at a functional level
- How the AWU state machine operates
- How to configure the register settings used by AWU
- How to fine-tune detection behavior for a specific use case
- How to integrate the wake-up flow on the host side
- How to evaluate the performance of the feature
- And which limitations and recommendations should be considered during integration

1 Acronyms and abbreviations

Table 1. Acronyms and abbreviations

Acronym/abbreviation	Definition
AE	auto exposure
AWU	auto wake-up
CSI	camera serial interface
DUT	device under test
FPS	frames per second
FoV	field of view
GPIO	general-purpose input/output
HW_STANDBY	hardware standby
I ² C	inter-integrated circuit
I3C	improved inter-integrated circuit
ISL	intelligent status line
MCU	microcontroller unit
MIPI	mobile industry processor interface
NVM	nonvolatile memory
ROI	region of interest
SPI	serial peripheral interface
STBY	standby
SW_STANDBY	software standby
ULP	ultralow power

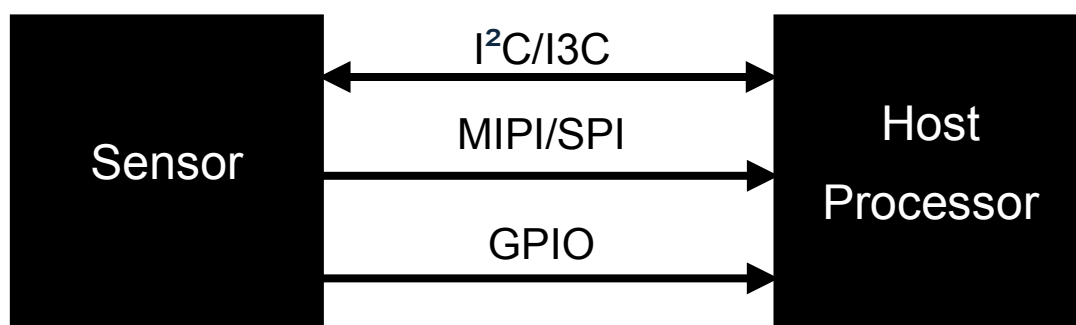
2 Functional description

The auto wake up (AWU) feature enables low-power scene monitoring without continuously streaming image data. From a system point of view, AWU allows the host processor to remain in a low-power or idle state while the sensor monitors the scene autonomously. When a relevant change is detected, the sensor notifies the host through a GPIO interrupt, so the host can resume processing and capture the image stream only when needed.

In a typical integration, the sensor and host are connected through:

- I²C for sensor configuration and status monitoring
- MIPI CSI-2, SPI, or I3C for image streaming in normal mode
- GPIO for the AWU interrupt notification

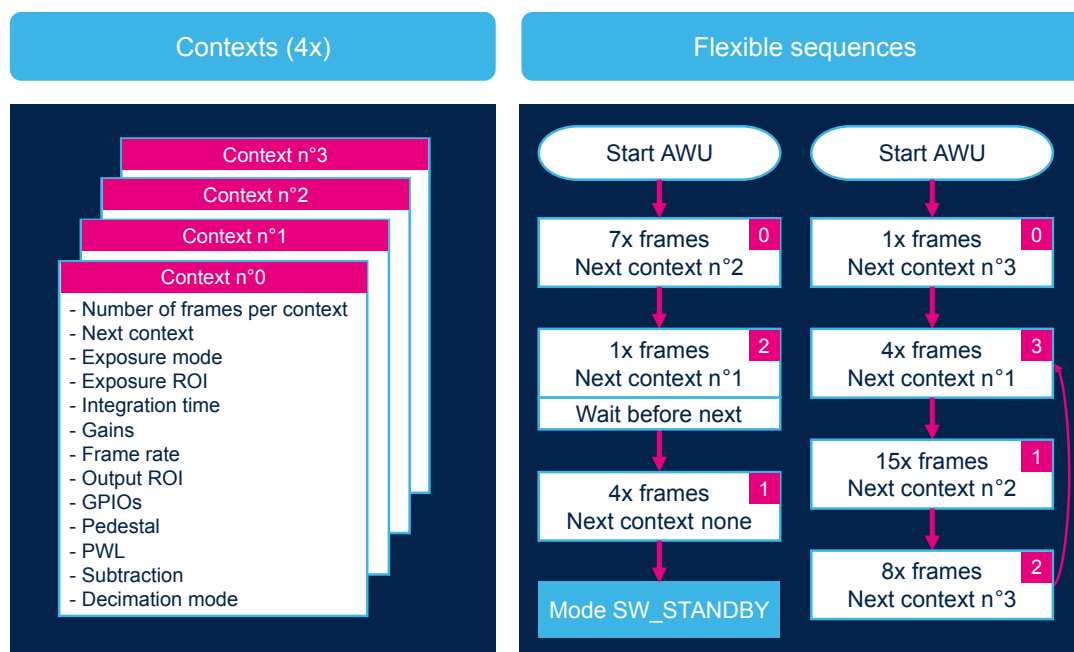
Figure 1. Typical integration overview



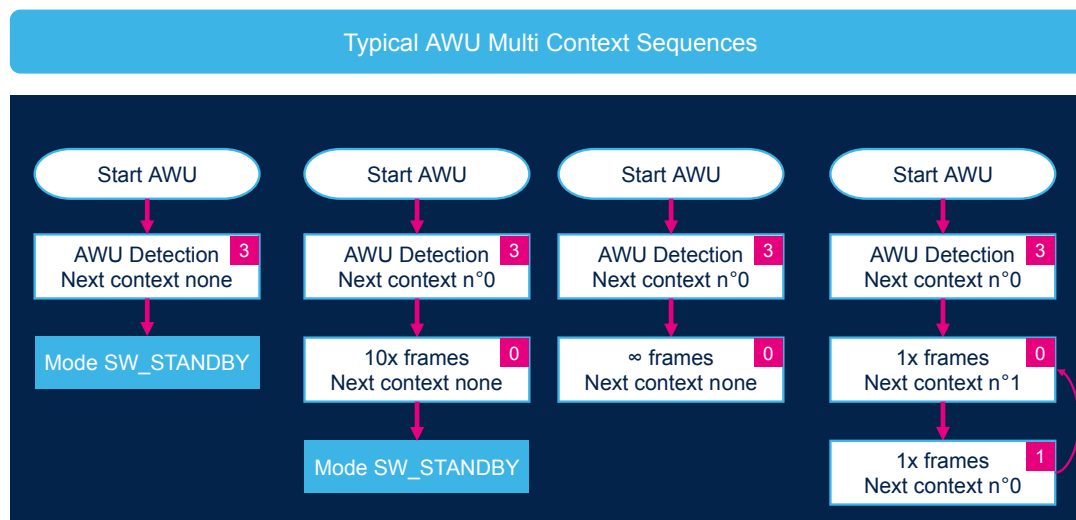
2.1 System overview

The sensor allows the configuration of up to four different frame-setup contexts. These contexts can be chained one to another in an ending or nonending sequence, depending on the application requirement.

Figure 2. Multicontext sequence overview



For AWU operation, the context three is used as the monitoring context and monitoring the scene while the host remains idle. When the sensor detects a wake-up event, it follows the programmed context chain. Then, it transitions to the next context. If no next context is assigned, it returns to standby. This process is shown in Figure 3. Typical AWU multicontext sequences.

Figure 3. Typical AWU multicontext sequences


The detailed context mechanism and the corresponding AWU register settings are described in [Section 3: AWU configuration and tuning](#).

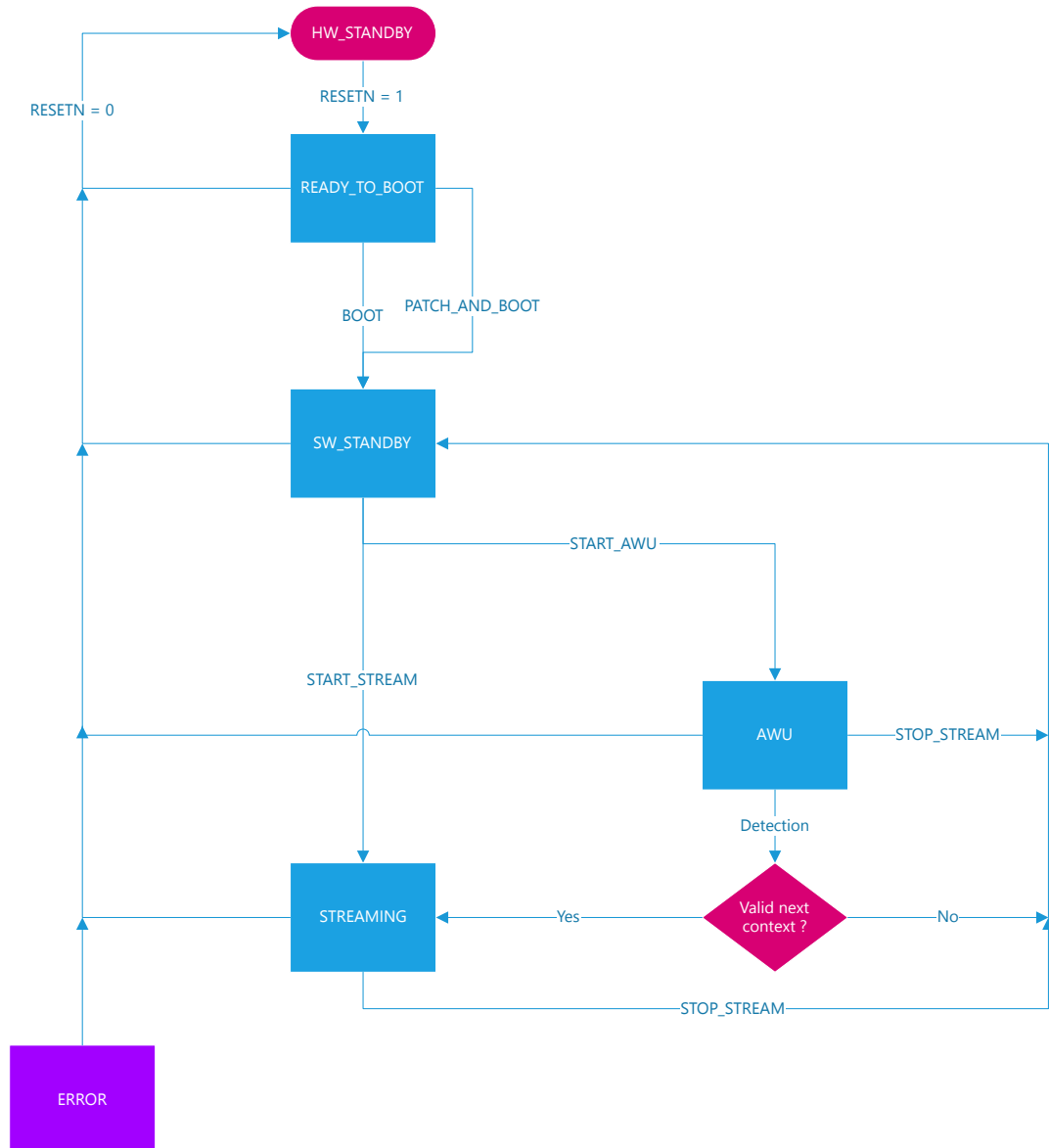
According to the sensor's firmware state machine, AWU is started from SW_STANDBY using the START_AWU command.

The main firmware states involved in the AWU use case are listed below:

- HW_STBY
- READY_TO_BOOT
- SW_STANDBY
- STREAMING_AWU_MODE
- STREAMING
- ERROR

A typical flow is:

1. The device is powered up and kept in HW_STBY while RESETN is low
2. The device enters READY_TO_BOOT when RESETN is released
3. The sensor is booted and placed in SW_STANDBY
4. The host starts AWU using START_AWU
5. The sensor runs in AWU state until a wake-up event occurs
6. After the wake-up event triggers, the sensor firmware checks the next context in the context chain. It uses the CONTEXT_NEXT_CONTEXT.CTX3 register to determine whether it is valid.
 - If the value is greater than three, the next context is considered invalid, and the sensor returns to SW_STANDBY
 - If the value is less than or equal to three, the next context is considered valid, and the sensor continues streaming using the configuration of the selected context

Figure 4. Firmware state machine


2.2 AWU operating principle

During AWU mode, the sensor uses the context three as the dedicated monitoring context. The AWU algorithm first performs a convergence phase to stabilize the reference used for the wake-up detection. Once convergence is complete, the sensor enters the detection phase, in which the current scene statistics are compared with the learned reference.

To report the wake-up event, the GPIO used by the monitoring context must be configured as AWU_DETECTION. Alternatively, if INTERRUPT_MODE_TO_HOST is set to I3C_SLAVE_IBI in I3C_FRAME_READOUT_CTRL at address 0x0324, the wake-up event is reported over I3C instead of the GPIO interrupt line.

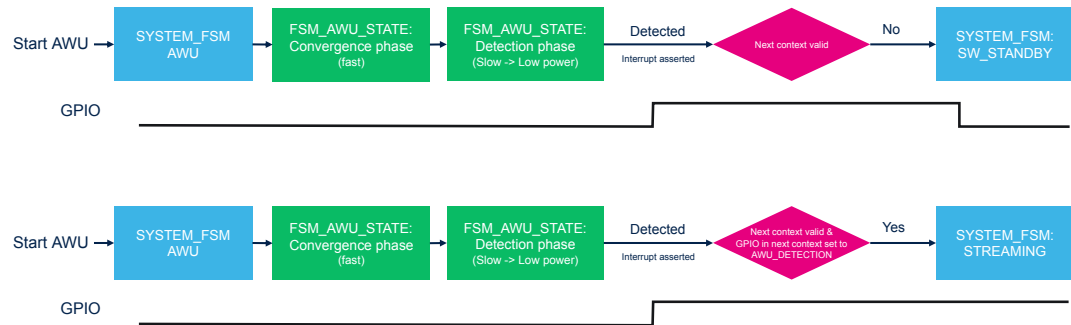
If the programmed next context is valid, the sensor transitions to that context after detection. Otherwise, it returns to SW_STANDBY.

If the number of zones that exceed the detection criterion reaches the programmed threshold, the AWU event is generated and the sensor asserts the configured interrupt output.

The interrupt output on the GPIO depends on the context 3 GPIO configuration (GPIO_0_CTRL/GPIO_1_CTRL/GPIO_2_CTRL/GPIO_3_CTRL), including polarity. The GPIO state remains latched until one of the following conditions is met:

- The sensor returns to the SW_STANDBY state
- The sensor switches to the next context while the GPIO mode is not configured as AWU_DETECTION

Figure 5. GPIO sequence in AWU mode



2.3 Monitoring AWU status

To help the host monitor the progress of the AWU flow, the sensor exposes dedicated status registers. These registers provide visibility both on the device-level firmware state and on the internal AWU algorithm state.

The following registers are used to observe the status of AWU operation:

- SYSTEM_FSM at address 0x001C
- FSM_AWU_STATE at address 0x012F

The key difference between these two states is shown in the table below:

Table 2. Register for AWU state

Register	Scope	What it indicates to the host
SYSTEM_FSM	Device-level	Overall sensor state, such as standby, streaming, or error
FSM_AWU_STATE	AWU algorithm-level	Current AWU progress, such as convergence or detection

FSM_AWU_STATE reports the internal AWU algorithm progress:

- AWU_OFF
- AWU_CONVERGENCE_RUNNING
- AWU_CONVERGENCE_DONE
- AWU_DETECTION_RUNNING
- AWU_DETECTION_DONE
- AWU_ERROR

SYSTEM_FSM reports the overall device state:

- HW_STBY
- READY_TO_BOOT
- SW_STANDBY
- STREAMING
- STREAMING_AWU_MODE
- FW_STALL
- ERROR

In practice, FSM_AWU_STATE is used to confirm whether the sensor has completed convergence and whether it is currently monitoring for a wake-up event. SYSTEM_FSM is used to confirm whether the sensor is in the correct device state to start AWU, to start streaming, or to recover from an error condition.

2.4 AWU phases

The AWU algorithm is organized into two main phases:

- **Convergence:**
During convergence, the sensor builds the reference used for the wake-up decision.
- **Detection:**
During detection, the sensor compares the current scene statistics to the learned reference. It checks whether the programmed detection condition is met.

Important: The equations in [Section 2.4.1: Convergence phase](#) and [Section 2.4.2: Detection phase](#) are simplified illustrative expressions. They are provided to help explain the AWU principle and do not represent the full internal implementation.

2.4.1 Convergence phase

During convergence, the sensor captures several frames to compute the reference levels used by the detection engine. A simplified expression for the learned reference of one zone is:

$$S_{ref,Z} = \frac{1}{N} \sum_{n=0}^{N-1} (S_{pixel,Z}(n) - P)$$

Where:

- $S_{ref,Z}$ is the learned reference level for zone Z
- $S_{ref,Z}(n)$ is the raw accumulated statistic measured in zone Z on frame n
- P is the pedestal term
- N is the number of frames

The convergence frame rate mainly affects the time required to complete this phase. A higher convergence frame rate reduces the time needed to enter the detection phase, but it does not change the detection behavior once convergence is complete.

2.4.2 Detection phase

Once the convergence phase has finished, the sensor enters the detection phase. In this phase, the current scene statistics of each zone is compared with the reference learned for the same zone.

A simplified way to express the detection condition for one zone is:

$$|S_{current,Z} - S_{ref,Z}| > T_{eff}$$

Where:

- $S_{current,Z}$ is the current scene statistic of zone Z
- $S_{ref,Z}$ is the learned reference level of the same zone
- T_{eff} is an effective threshold derived from the programmed AWU_DETECTION_THRESHOLD and the internal detection processing.

In other words, each zone is evaluated independently against its own learned reference, and a zone is considered changed when the difference exceeds the effective threshold.

3 AWU configuration and tuning

This section describes the main registers used to configure and fine-tune the AWU feature. The AWU configuration described in this note uses context three. Unless otherwise stated, the settings in this section refer to context three.

3.1 Configuration summary

The [Table 3. Registers link to AWU](#) summarizes the main AWU-related registers used in this note. For the full register definition and bitfields, refer to the user manual.

Table 3. Registers link to AWU

Function	Register group	Register	Address	Notes
AWU status monitor	STATUS	AWU_STATUS	0x0144	AWU detection status and detection per zones
Start AWU command	CMD	STBY.CMD	0x0201	Use START_AWU software from standby
Main AWU control	STREAM_STATICS	AWU_CTRL	0x036C	Main AWU configuration register
Detection threshold	STREAM_STATICS	AWU_DETECTION_THRESHOLD	0x0370	Sensitivity of the zone comparison
Max integration	STREAM_STATICS	EXPOSURE_USER_MAX_COARSE_INTEGRATION_LINES	0x0372	Limits maximum integration time
AWU power controls	STREAM_STATICS	AWU_IPS_CTRL	0x0384	Controls bypass of image-processing blocks
Active zones	STREAM_CTX3	EXPOSURE_STATS_ACTIVE_ZONE	0x0608	Zone mask used by AWU in context 3
Frame length	STREAM_CTX3	FRAME_LENGTH	0x05FC	Frame length used by AWU in context 3
Exposure mode	STREAM_CTX3	EXPOSURE_MODE	0x05F0	Exposure mode used by AWU in context 3
GPIO output	STREAM_CTX3	GPIO_x_CTRL	0x060D to 0x0610	Set GPIO mode to AWU_DETECTION in context 3

AWU_IPS_CTRL can be used in AWU mode to bypass image-processing functions such as dark calibration, defect correction, and noise reduction, to reduce sensor-side power consumption.

3.2 Convergence phase configuration

The convergence phase is the period during which the sensor stabilizes exposure and builds the reference data used by the detection engine.

Relevant register:

- AWU_CTRL.FRAME_LENGTH_FOR_CONVERGENCE
 - Register at address 0x036C [31:16]
 - AWU convergence frame rate

A higher convergence frame rate allows the sensor to become ready for detection faster. It shortens the time needed for the convergence phase. However, it does not significantly change the detection logic that is used after convergence.

Other settings that influence convergence include:

- The exposure mode
- The cold-start exposure configuration
- And whether the monitored scene is stable during convergence

Recommended actions

- Do keep the scene stable during convergence
- Do use a short convergence time when the installation environment is repeatable
- Do not move the sensor or the target while convergence is running
- Do not assume that convergence is valid if the scene changes abruptly during this phase

3.3

Detection threshold

The detection threshold defines how much change is required before a zone is considered changed. A lower threshold increases sensitivity, while a higher threshold improves robustness against small variations in illumination or background motion.

Relevant register:

- AWU_DETECTION_THRESHOLD
 - Register at address 0x0370 [15:0]
 - Typical tuning range: 1 - 50

Start from the default value and tune the threshold gradually based on the target scene, mounting geometry, lens characteristics, and lighting stability.

Figure 6. Threshold set to 6



Figure 7. Threshold set to 255



Note:

Please note the following information:

- Small objects passing in the background may be detected with a low threshold
- Large changes such as a strong light transition are more likely to trigger detection even with a higher threshold
- Threshold tuning becomes especially important when the object's brightness is close to the background brightness

Recommended actions

- Do start with the default threshold and decrease gradually if real events are missed
- Do increase the threshold if false wakeups are observed
- Do not use an overly low threshold in a scene with unstable illumination
- Do not assume a single threshold value works for every lens, mounting height, or lighting condition

3.4 Number of zones required to wake up

The image is divided into a 4 x 4 grid. Statistics are computed and compared zone by zone. The AWU event is asserted only when the number of detected zones reaches the programmed threshold.

Relevant register:

- AWU_CTRL.NO_OF_ZONE_TO_WAKEUP
 - Register at address 0x036C [15:11]
 - Typical range: 1 - 16

During development, a high valid threshold can be used to keep the sensor in AWU mode during aggressive test scenarios. Even when the wake-up condition is not met, AWU_STATUS still indicates which zones exceeded the detection threshold. This makes it easier to assess the spatial extent of the scene change before the final AWU trigger is reached.

The number of zones is useful for distinguishing between:

- Local motion, and
- More global scene changes

For example, a movement close to the image sensor can affect more zones than a far-away person walking in the background.

Figure 8. Movement close to the camera module triggers four zones

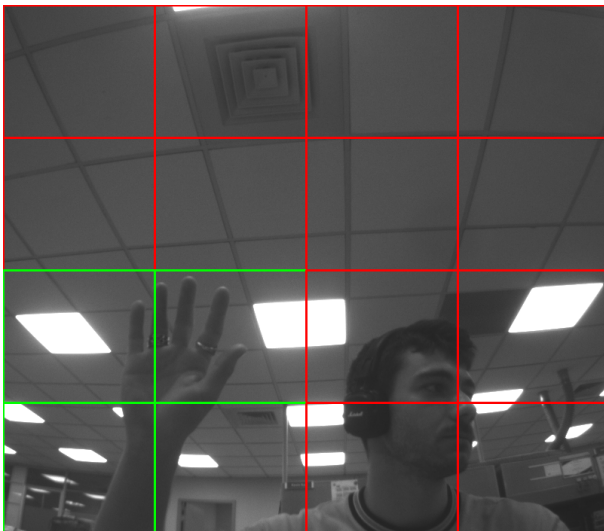
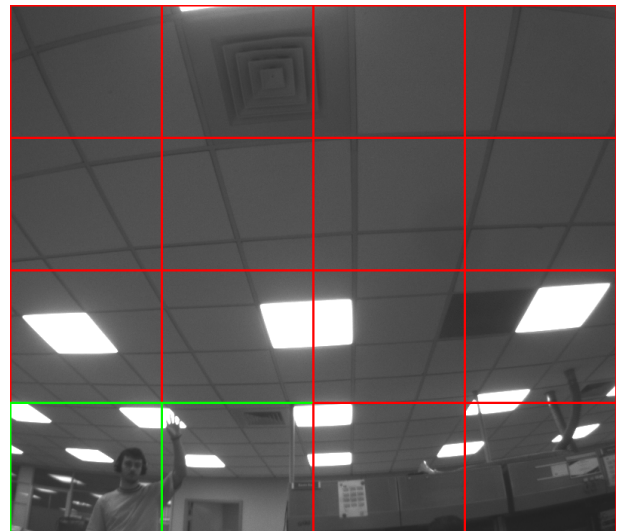


Figure 9. Movement in the background only triggers two zones



Recommended actions

- Do increase the number of zones if you want to ignore small local changes
- Do decrease the number of zones if you want to detect small but important movement
- Do not rely on a single-zone trigger if the scene is very noisy
- Do not set the number of zones so high that valid motion is systematically ignored

3.5 Zone selection

The following subsections describe the zone layout, the active-zone mask, and the interaction between AWU and AE zone mapping.

3.5.1 Zone layout and numbering

The monitored area is divided into a 4 x 4 grid. Zone IDs are assigned in raster order from left to right and from top to bottom. The zone numbering follows the oriented output image after the sensor flip settings are applied. For the zone layout:

- Zone 0 is the top-left zone
- Zone 3 is the top-right zone
- Zone 4 is the first zone of the second row
- Zone 15 is the bottom-right zone, as shown in the figure

Figure 10. Zone mapping of nonflipped image

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Figure 11. Zone mapping of flipped image

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

The same zone numbering is used by the ZONE_DETECTION field in the AWU_STATUS register at address 0x0144[15:0], bit 0 maps to zone 0 and bit 15 maps to zone 15, as shown in the following figure:

Figure 12. Zone and register mapping for AWU

Zone 0 Bit 15	Zone 1 Bit 14	Zone 2 Bit 12	Zone 3 Bit 12
Zone 4 Bit 11	Zone 5 Bit 10	Zone 6 Bit 9	Zone 7 Bit 8
Zone 8 Bit 7	Zone 9 Bit 6	Zone 10 Bit 5	Zone 11 Bit 4
Zone 12 Bit 3	Zone 13 Bit 2	Zone 14 Bit 1	Zone 15 Bit 0

3.5.2 Active zone mask

The monitored area can be restricted with EXPOSURE_STATS_ACTIVE_ZONE. This is useful when the application needs to detect motion only in a specific part of the field of view and ignore motion elsewhere. The selected mask should be verified after applying flip settings, because the zone mapping follows the oriented output image.

Relevant register:

- EXPOSURE_STATS_ACTIVE_ZONE
 - Register at address 0x0608 [15:0]

Figure 13. Movement from the active zone does not trigger AWU

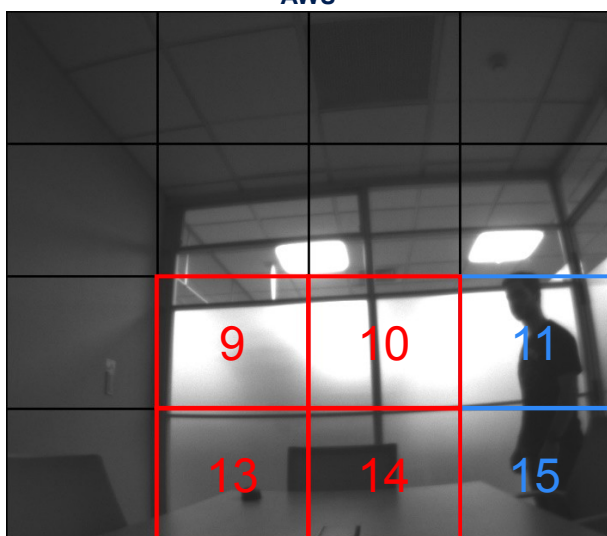
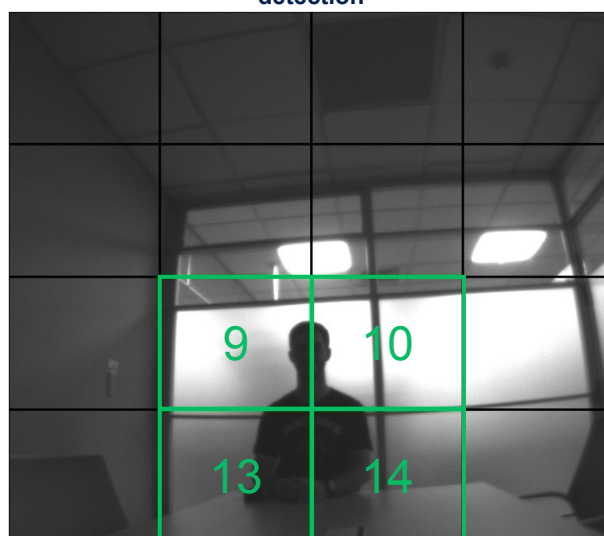


Figure 14. Movement within the active zone is used for detection



Important: If you need precise detection by selecting a narrow zone and by using a tight wake-up criterion, the system may behave more predictably with manual exposure instead of auto exposure. In AWU mode, a change in one zone can influence exposure behavior in the rest of the image and may cause zones without motion to be affected.

Figure 15. Manual exposure mode

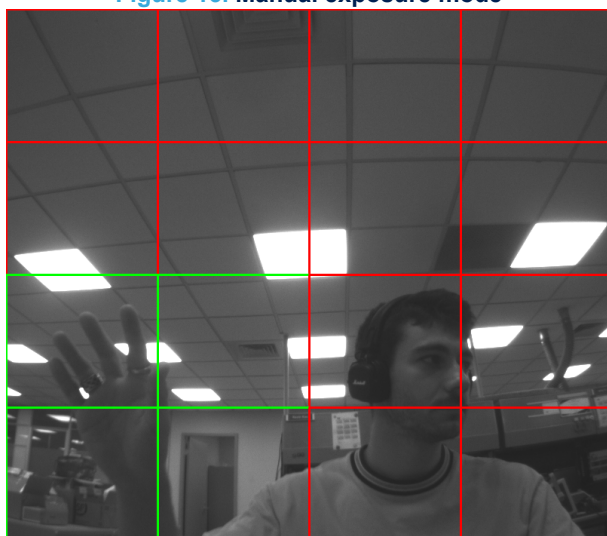


Figure 16. Auto exposure mode



Recommended actions

- Do restrict the active zones if the use case has a clear region of interest
- Do align the selected zones with the actual target area
- Do not expect perfect physical localization from exposure-based wake-up alone
- Do not assume that only the moving zone is affected when AE is enabled

3.5.3 AWU and AE zone-mapping limitation

A mismatch may occur between the AWU detection mask and the auto exposure statistics ROI. In such cases, AWU may monitor one region of the image while AE computes statistics and adjusts exposure using another region. This can lead to unexpected behavior in asymmetric scenes.

This limitation applies only when auto exposure is enabled. In manual exposure mode, the issue does not apply because exposure settings are fixed and no auto exposure statistics update is performed.

Recommendation

To reduce the risk when AWU and AE are both enabled, use a spatial symmetric active zone mask centered on the image. In this note, a spatial symmetry mask means an EXPOSURE_STATS_ACTIVE_ZONE configuration that remains symmetric around the image center and helps keep AWU and AE focused on comparable physical areas.

- Full-frame mask: EXPOSURE_STATS_ACTIVE_ZONE = 0xFFFF
- Center 2 columns across all rows: EXPOSURE_STATS_ACTIVE_ZONE = 0x6666
- Center 2 rows: EXPOSURE_STATS_ACTIVE_ZONE = 0x0FF0
- Center 2 x 2 zones: EXPOSURE_STATS_ACTIVE_ZONE = 0x0660

This approach helps avoid a situation where AWU detection zones and AE statistics zones are not linked to the same physical region.

Best practices

- Do use a spatial symmetry mask when AWU and AE are both enabled
- Do verify the zone mapping after applying flip or orientation settings
- Do not assume that the AWU detection mask and AE statistics ROI will always refer to the same physical area
- Do not use a narrow asymmetric ROI unless the interaction between AWU and AE has been validated in the target scene

3.6 Frame rate during detection

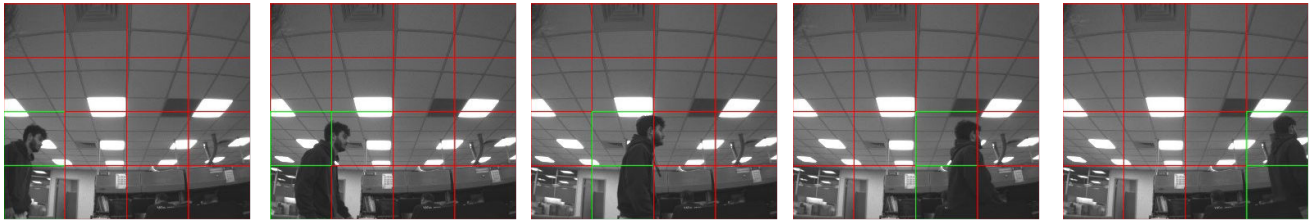
The detection phase usually runs at the lowest practical frame rate that satisfies the system wake-up latency and motion-detection requirements. A lower frame rate reduces power consumption, but it also increases the risk of missing fast motion. Choose the detection frame rate according to the expected motion speed, host wake-up latency, and power budget.

Relevant registers:

- STREAM_CTX3.FRAME_LENGTH
 - Register at address 0x05FC [31:0]
 - Typical range: 3 - 20 fps
 - Most of the time, you want to run the AWU with the lowest frame rate to save power
 - If the FPS is too low and the movement too fast, you might not trigger the AWU

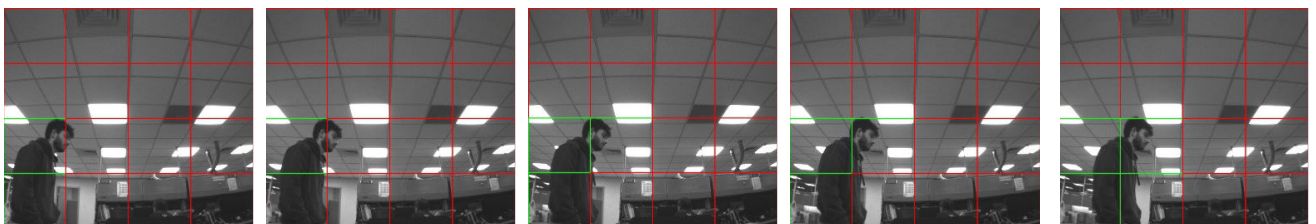
Note: When the AWU detection frame rate is set to 3 fps and NO_OF_ZONE_TO_WAKEUP is programmed to 4, fast motion may be missed, as shown in the following figure.

Figure 17. Detection frame rate effect: 3 fps



- AWU running at 20 fps with the same number of zones is more likely to trigger reliably, as shown in the following figure

Figure 18. Detection frame rate effect: 20 fps



Recommended actions

- Do reduce detection FPS when power is the primary priority
- Do increase detection FPS when fast motion must be detected
- Do not reduce the frame rate so much that the target crosses the scene between two frames
- Do not use the same FPS for convergence and detection unless it is justified by your use case

3.7 Configuration summary table

Table 4. AWU configuration summary

Parameter	Recommended use	Effect	Notes / Register address
AWU_CTRL.FRAME_LENGTH_FOR_CONVERGENCE	Tune convergence speed	Faster or slower readiness	0x036C
AWU_DETECTION_THRESHOLD	Tune sensitivity	Lower = more sensitive, higher = more robust	0x0370
AWU_CTRL.NO_OF_ZONE_TO_WAKEUP	Tune decision strictness	More zones = less false wake-ups, less sensitivity	0x036C
EXPOSURE_STATS_ACTIVE_ZONE	Restrict monitored area	Focus detection on a target region	0x0518
FRAME_LENGTH	Tune detection frame rate	Power vs responsiveness trade-off	Context-dependent 0x05FC
EXPOSURE_MODE	Choose AE or manual	Affects detection stability	Context-dependent 0x05F0
AWU_IPS_CTRL	Control image-processing path	Can reduce processing overhead	0x0384

4 Host integration and step-by-step AWU flow

4.1 Host-side overview

The host is responsible for:

- Starting AWU from software standby
- Waiting for the wake-up interrupt
- Switching the sensor to full streaming mode
- And optionally re-entering AWU after the event has been processed

4.2 Step-by-step host firmware flow

- **Step 1: power up the device**
 - Provide the external supplies and clock according to the datasheet and user manual
- **Step 2: bring the device out of hardware standby**
 - Set RESETN high and wait until the device reaches READY_TO_BOOT
- **Step 3: configure the sensor for AWU**
 - Program the AWU-related registers, including:
 - AWU_CTRL
 - AWU_DETECTION_THRESHOLD
 - EXPOSURE_STATS_ACTIVE_ZONE
 - EXPOSURE_MODE
 - FRAME_LENGTH
 - GPIO configuration for AWU_DETECTION
- **Step 4: start AWU**
 - Issue START_AWU through STBY.CMD at 0x0201
- **Step 5: wait for convergence to complete**
 - Optionally monitor FSM_AWU_STATE at 0x012F until convergence is complete
- **Step 6: confirm that AWU is active**
 - Once convergence is complete, the sensor enters the AWU detection state and waits for a valid event
 - Optionally check that FSM_AWU_STATE at 0x012F is AWU_DETECTION_RUNNING
- **Step 7: handle the GPIO interrupt**
 - When the wake-up event occurs, the configured GPIO line asserts an interrupt to the host
 - Read AWU_STATUS at 0x0144 to confirm that AWU has been triggered and to identify the zones that caused the wake-up
- **Step 8: transition to normal streaming**
 - The host starts the normal stream using the standard streaming command path
- **Step 9: process the frame data**
 - The host retrieves and processes image data through MIPI/I3C/SPI
- **Step 10: return to AWU if needed**
 - If the application is event-driven and low-power operation is required continuously, the host can reconfigure and restart the AWU after processing

4.3 Relevant registers

Section 3.1: **Configuration summary** lists the full AWU register summary and tuning knobs.

This section highlights only the registers and command that the host must:

- Use to configure AWU
- Start the feature
- And monitor its status during execution

Table 5. Required host actions for AWU

Action	Register Group	Register	Address	Note
Set	STREAM_STATICS	AWU_CTRL	0x036C	Main AWU configuration
Set	STREAM_STATICS	AWU_DETECTION_THRESHOLD	0x0370	Sensitivity tuning
Set	STREAM_CTX3	EXPOSURE_MODE	0x05F0	Manual exposure or auto exposure mode in context 3
Set	STREAM_CTX3	EXPOSURE_STATS_ACTIVE_ZONE	0x0608	Active zones / ROI in context 3
Set	STREAM_CTX3	GPIO_X_CTRL	0x060D to 0x0610	Event report mode in context 3
Command	CMD	STBY	0x0201	Start AWU from SW_STANDBY, by assigning START_AWU
Read /Check	STATUS	SYSTEM_FSM	0x001C	System FSM status
Read /Check	STATUS	FSM_AWU_STATE	0x012F	AWU algorithm status
Read /Check	STATUS	AWU_STATUS	0x0144	AWU detection status and triggered zones.

5 Reference code

The reference code can be found in:

- STSW-IMG507 (vd55g1_auto_wake_up.py)
- STSW-VD65G4 (vd55g1.c::vd55g1_start_awu)

For clarity, this application note may use a pseudocode instead of binding the description to a specific demo implementation.

5.1 Recommended operating scenarios

Depending on the host capability after interrupting reception, the application can adopt one of the following operation scenarios:

1. Detection triggered, then stop:
The sensor detects a scene change and stops streaming. The host starts the next image-streaming session manually.
2. Detection triggered, then keep streaming for a few frames:
The sensor detects a scene change and continues streaming for a limited number of frames before stopping. This scenario is useful when the host needs a short image burst after the trigger.
3. Detection triggered, then keep streaming indefinitely:
The sensor detects a scene change and continues streaming until the host explicitly stops it. This scenario is useful when the wake-up event is only used to exit the low-power monitoring mode and the host is able to keep the image pipeline running.

The selection of the operating scenario depends on the programmed context chaining and repeat-count configuration, especially the values of CONTEXT_NEXT_CONTEXT and CONTEXT_REPEAT_COUNT_CTX0 / CONTEXT_REPEAT_COUNT_CTX1 / CONTEXT_REPEAT_COUNT_CTX2 / CONTEXT_REPEAT_COUNT_CTX3.

Generally, if the host cannot immediately resume image reception after interrupting reception, it is recommended to use the first scenario. If the host can resume image reception directly, the second or third scenario can be used depending on whether a limited burst or continuous streaming is desired.

5.2 Reference pseudocode

The following pseudocode illustrates a typical host-side implementation flow. The actual API names depend on the software stack used by the host platform.

5.2.1 Common configuration primitives

```
function set_default_configuration():
    write_register_32(SENSOR_SETTINGS_EXT_CLOCK, 12000000)
    write_register_32(SENSOR_SETTINGS_MIPI_DATA_RATE, 1200000000)
    write_register_16(STATICS_LINE_LENGTH, 1344)
    write_register_16(STATICS_FORMAT_CTRL, RAW8)
    write_register_16(STATICS_ORIENTATION, X_FLIP)
    write_register_8(DYNAMICS_EXPOSURE_TARGET_PERCENTAGE_A, 50)
    write_register_16(STATICS_CONTEXT_REPEAT_COUNT_CTX0, 0)
    write_register_16(STATICS_CONTEXT_REPEAT_COUNT_CTX1, 0)
    write_register_16(STATICS_CONTEXT_REPEAT_COUNT_CTX2, 0)
    write_register_16(STATICS_CONTEXT_REPEAT_COUNT_CTX3, 0)

    for c in range(4):
        write_register_16(CONTEXTS_X_START + c * context_size, 1)
        write_register_16(CONTEXTS_Y_START + c * context_size, 2)
        write_register_16(CONTEXTS_X_WIDTH + c * context_size, 800)
        write_register_16(CONTEXTS_Y_HEIGHT + c * context_size, 700)
        write_register_16(CONTEXTS_DARKCAL_PEDESTAL + c * context_size, 0)
        # Toggle GPIO2 once motion is detected
        write_register_8(CONTEXTS_GPIO_2_CTRL, 0xD)

function configure_auto_wake_up():
    # Constants for AWU configuration
    awu_zone_count = 2
    awu_context3_offset = 3 * context_size
```

```

disable_frame_output_during_detection = 1

set_default_configuration()

# Static configuration settings for AWU
write_register_16(STATICS_EXPOSURE_USER_MAX_COARSE_INTEGRATION_LINES, 401)
write_register_16(STATICS_AWU_DETECTION_THRESHOLD, 6 << 8)
write_register_16(STATICS_AWU_CTRL, default_awu_ctrl_value + (awu_zone_count << 11))
write_register_16(STATICS_AWU_FRAME_LENGTH_FOR_CONVERGENCE, 1673)
write_register_16(STATICS_CONTEXT_NEXT_CONTEXT, 0xFFFF)

# Context 3 configuration for AWU
write_register_16(CONTEXT_FRAME_LENGTH + awu_context3_offset, 22160)
write_register_8(CONTEXT_MASK_FRAME_CTRL + awu_context3_offset, disable_frame_output_during_detection)
write_register_8(CONTEXT_READOUT_CTRL + awu_context3_offset, SUB_SAMPLING_X2)

# Start AWU
write_register_8(CMD_STBY, START_AWU)

```

5.2.2 Scenario 1: detection triggered, then stop

```

function awu_detection_then_stop_configuration():
    set_default_configuration()
    configure_auto_wake_up()
    write_register_16(STATICS_CONTEXT_REPEAT_COUNT_CTX0, 0)
    write_register_16(STATICS_CONTEXT_REPEAT_COUNT_CTX1, 0)
    write_register_16(STATICS_CONTEXT_REPEAT_COUNT_CTX2, 0)
    write_register_16(STATICS_CONTEXT_REPEAT_COUNT_CTX3, 0)
    write_register_16(STATICS_CONTEXT_NEXT_CONTEXT, 0xFFFF)

function awu_event_handler():
    _wait_for_interrupt()
    system_fsm = read_register_16(STATUS_SYSTEM_FSM)
    awu_status = read_register_16(STATUS_AWU_STATUS)
    if system_fsm == STREAMING_AWU_MODE_WAITING:
        keep_sleep()
        return
    elif system_fsm == SW_STANDBY:
        if awu_status != 0:
            display_message("Sensor stopped, event detected")
            scene_change_check(awu_status)
        else:
            display_message("Sensor stopped, no event detected")
            write_register_8(CMD_STBY, START_AWU)
            display_message("AWU restarted, waiting for an event")
            return

function scene_change_check(awu_status):
    awu_detected = awu_status.detection
    zones = awu_status.zone_detection

    write_register_8(CMD_STBY, START_STREAM)
    metadata, image = get_image()
    target_detected = object_detection(metadata, image)

    if awu_detected and target_detected:
        display_message("Scene change detected, target detected")
        configure_sensor_for_normal_streaming()
    elif awu_detected and not target_detected:
        display_message("Scene change detected, no target detected (false alarm)")
        write_register_8(CMD_STBY, START_AWU)
        display_message("Fall back to AWU, waiting for an event")
        keep_sleep()

```

5.2.3 Scenario 2: detection triggered, then keep streaming for 10 frames

```
function awu_detection_then_10frames_then_stop_configuration():
    set_default_configuration()
    configure_auto_wake_up()
    write_register_16(STATICS_CONTEXT_REPEAT_COUNT_CTX0, 10)
    write_register_16(STATICS_CONTEXT_REPEAT_COUNT_CTX1, 0)
    write_register_16(STATICS_CONTEXT_REPEAT_COUNT_CTX2, 0)
    write_register_16(STATICS_CONTEXT_REPEAT_COUNT_CTX3, 0)
    write_register_16(STATICS_CONTEXT_NEXT_CONTEXT, 0x0FFF)

function awu_event_handler():
    _wait_for_interrupt()
    system_fsm = read_register_16(STATUS_SYSTEM_FSM)
    awu_status = read_register_16(STATUS_AWU_STATUS)
    if system_fsm == STREAMING_AWU_MODE_WAITING:
        keep_sleep()
        return
    elif system_fsm == STREAMING:
        frame_list = get_10_frames()
        metadata, image = image_processing(frame_list)
        target_detected = object_detection(metadata, image)
        awu_detected = awu_status.detection
        zones = awu_status.zone_detection
        if awu_detected and target_detected:
            display_message("Scene change detected, target detected")
            reconfigure_sensor_for_normal_streaming()
        elif awu_detected and not target_detected:
            display_message("Scene change detected, no target detected (false alarm)")
            write_register_8(CMD_STBY, START_AWU)
            display_message("Fall back to AWU, waiting for an event")
            keep_sleep()
```

5.2.4 Scenario 3: detection triggered, then keep streaming indefinitely

```
function awu_detection_then_streaming_configuration():
    set_default_configuration()
    configure_auto_wake_up()
    write_register_16(STATICS_CONTEXT_REPEAT_COUNT_CTX0, 0)
    write_register_16(STATICS_CONTEXT_REPEAT_COUNT_CTX1, 0)
    write_register_16(STATICS_CONTEXT_REPEAT_COUNT_CTX2, 0)
    write_register_16(STATICS_CONTEXT_REPEAT_COUNT_CTX3, 0)
    write_register_16(STATICS_CONTEXT_NEXT_CONTEXT, 0x0FFF)

function awu_event_handler():
    _wait_for_interrupt()
    system_fsm = read_register_16(STATUS_SYSTEM_FSM)
    awu_status = read_register_16(STATUS_AWU_STATUS)
    if system_fsm == STREAMING_AWU_MODE_WAITING:
        keep_sleep()
        return
    elif system_fsm == STREAMING:
        metadata, image = get_image()
        target_detected = object_detection(metadata, image)
        awu_detected = awu_status.detection
        zones = awu_status.zone_detection
        if awu_detected and target_detected:
            display_message("Scene change detected, target detected")
            write_register_8(CMD_STREAMING, STOP_STREAM)
            reconfigure_sensor_for_normal_streaming()
        elif awu_detected and not target_detected:
            display_message("Scene change detected, no target detected (false alarm)")
            write_register_8(CMD_STBY, START_AWU)
            display_message("Fall back to AWU, waiting for an event")
            keep_sleep()
```

5.3 Streaming behavior and context chaining after AWU detection

- The CONTEXT_NEXT_CONTEXT programming determines whether the sensor returns to SW_STANDBY or continues streaming after AWU detection
- The actual behavior depends on the context chain programmed by the host and on the repeat-count configuration of the selected context
- The pseudocode above is intentionally simplified. It is meant to illustrate the host-side control flow, not to represent the exact SDK or firmware implementation

6 Limitations and recommendations

6.1 Scene stability during convergence

The convergence phase assumes that the monitored scene is stable. Large scene changes during convergence can prevent the reference values from stabilizing correctly and may reduce the reliability of the subsequent detection phase.

6.2 Choosing between auto exposure and manual exposure

AE mode is convenient and generally works well, but exposure adaptation can change the spatial behavior of the trigger. When stronger control over the wake-up region is required, manual exposure provides more deterministic behavior because the exposure settings remain fixed.

Refer to [Section 3.5.3: AWU and AE zone-mapping limitation](#) when both features use different regions.

6.3 Low frame rate versus fast motion

A lower frame rate reduces power consumption, but it also increases the risk of missing fast motion. The detection frame rate should therefore be selected according to the required wake-up latency and the expected target motion speed.

6.4 Local motion versus the global scene changes

Small local changes may be missed if the threshold or the number of zones is set too conservatively. Conversely, overly aggressive settings can lead to false wakeups.

Large scene changes during convergence can prevent the reference values from stabilizing correctly, which may reduce detection reliability for some frames.

The tuning impact becomes more pronounced in low-contrast scenes, where the target brightness is close to the background brightness.

The results can vary significantly depending on the scene and may require scene-specific tuning.

6.5 Recommended actions

DO

- Do keep the scene stable during convergence
- Do lower the threshold if real events are missed
- Do raise the threshold if false wakeups occur
- Do use manual exposure when spatial precision matters
- Do validate the final configuration in the target scene

Do not

- Do not expect AE mode to give exact zone-localized behavior in every scene
- Do not run AWU at too low a frame rate if the target motion is fast
- Do not assume that one configuration fits all scenes

7 Terminology

Table 6. Terminology

Term	Definition
Auto wake-up (AWU)	A mechanism in the image sensor that automatically triggers a wake-up when a significant scene change is detected.
Frame per second (FPS)	The number of image frames captured or processed by the sensor per second.
Field of view (FoV)	The angular extent of the observable scene captured by the sensor and lens system.
Grid	Division of the sensor image into smaller regions for localized change detection.
Trigger accuracy	Percentage of times AWU correctly triggers upon actual scene changes.
False alarm rate	Percentage of times AWU triggers without actual scene change.
Miss detection rate	Percentage of times AWU fails to trigger despite actual scene change.
Response time	Time interval between the occurrence of a scene change and the AWU trigger signal.
Lux	A unit of illuminance.
Tilt angle	The angle at which the sensor is installed relative to the reference plane.
Intelligent Status Line (ISL)	Metadata lines inserted in the output stream.

Revision history

Table 7. Document revision history

Date	Version	Changes
13-Aug-2026	1	Initial release.

Contents

1	Acronyms and abbreviations	2
2	Functional description	3
2.1	System overview	3
2.2	AWU operating principle	5
2.3	Monitoring AWU status	6
2.4	AWU phases	7
2.4.1	Convergence phase	7
2.4.2	Detection phase	7
3	AWU configuration and tuning	8
3.1	Configuration summary	8
3.2	Convergence phase configuration	8
3.3	Detection threshold	9
3.4	Number of zones required to wake up	10
3.5	Zone selection	11
3.5.1	Zone layout and numbering	11
3.5.2	Active zone mask	12
3.5.3	AWU and AE zone-mapping limitation	13
3.6	Frame rate during detection	13
3.7	Configuration summary table	14
4	Host integration and step-by-step AWU flow	15
4.1	Host-side overview	15
4.2	Step-by-step host firmware flow	15
4.3	Relevant registers	15
5	Reference code	17
5.1	Recommended operating scenarios	17
5.2	Reference pseudocode	17
5.2.1	Common configuration primitives	17
5.2.2	Scenario 1: detection triggered, then stop	18
5.2.3	Scenario 2: detection triggered, then keep streaming for 10 frames	19
5.2.4	Scenario 3: detection triggered, then keep streaming indefinitely	19
5.3	Streaming behavior and context chaining after AWU detection	20
6	Limitations and recommendations	21
6.1	Scene stability during convergence	21
6.2	Choosing between auto exposure and manual exposure	21

6.3	Low frame rate versus fast motion	21
6.4	Local motion versus the global scene changes	21
6.5	Recommended actions	21
7	Terminology	22
Revision history		23
List of tables		26
List of figures		27

List of tables

Table 1.	Acronyms and abbreviations	2
Table 2.	Register for AWU state	6
Table 3.	Registers link to AWU	8
Table 4.	AWU configuration summary	14
Table 5.	Required host actions for AWU	16
Table 6.	Terminology	22
Table 7.	Document revision history	23

List of figures

Figure 1.	Typical integration overview	3
Figure 2.	Multicontext sequence overview	3
Figure 3.	Typical AWU multicontext sequences	4
Figure 4.	Firmware state machine.	5
Figure 5.	GPIO sequence in AWU mode	6
Figure 6.	Threshold set to 6	9
Figure 7.	Threshold set to 255	9
Figure 8.	Movement close to the camera module triggers four zones.	10
Figure 9.	Movement in the background only triggers two zones	10
Figure 10.	Zone mapping of nonflipped image	11
Figure 11.	Zone mapping of flipped image.	11
Figure 12.	Zone and register mapping for AWU	11
Figure 13.	Movement from the active zone does not trigger AWU	12
Figure 14.	Movement within the active zone is used for detection	12
Figure 15.	Manual exposure mode	12
Figure 16.	Auto exposure mode	12
Figure 17.	Detection frame rate effect: 3 fps	14
Figure 18.	Detection frame rate effect: 20 fps.	14

IMPORTANT NOTICE – READ CAREFULLY

STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice.

In the event of any conflict between the provisions of this document and the provisions of any contractual arrangement in force between the purchasers and ST, the provisions of such contractual arrangement shall prevail.

The purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST's terms and conditions of sale in place at the time of order acknowledgment.

The purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of the purchasers' products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

If the purchasers identify an ST product that meets their functional and performance requirements but that is not designated for the purchasers' market segment, the purchasers shall contact ST for more information.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to www.st.com/trademarks. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2026 STMicroelectronics – All rights reserved