AS3661

Programmable 9-channel LED Driver

General Description

The AS3661 is a 9-channel LED driver designed to produce lighting effects for mobile devices. A high efficiency charge pump enables LED driving over full Li- Ion battery voltage range. The device is equipped with an internal program memory, which allows operation without processor control. The AS3661 maintains excellent efficiency over a wide operating range by autonomously selecting the best charge pump gain based on LED forward voltage requirements. AS3661 is able to automatically enter power-save mode when LED outputs are not active, thus lowering idle current consumption down to 10µA (typ).

The AS3661 has an I^2C-compatible control interface with four pin selectable addresses. Also, the device has a flexible General Purpose Output (GPO), which can be used as a digital control pin for other devices. INT pin can be used to notify processor when a lighting sequence has ended (interrupt - function). Also, the device has a trigger input interface, which allows synchronization between multiple devices.The device requires only four small and low-cost ceramic capacitors.

The AS3661 is available in a tiny WL-CSP-25 (2.285 × 2.285mm) 0.4mm pitch package.

Ordering Information and Content Guide appear at end of datasheet.

Key Benefits & Features

The benefits and features of AS3661, Programmable 9-channel LED Driver are listed below:

Figure 1:
Added Value Of Using AS3661

Benefits	Features
Smooth light effects for the end user	• Three independent program execution engines • 9 programmable outputs with 25.5 mA full-scale current, 8- bit current setting resolution and 12-bit PWM control resolution
Long battery operating time	• Adaptive charge pump with 1x and 1.5x gain provides up to 95% LED drive efficiency
Easy system integration	• Charge pump with soft start and overcurrent/short • circuit protection
Easy system integration	• Built-in LED test
Long battery operating time	• Automatic power save mode; IVDD = 10 µA (typ.)

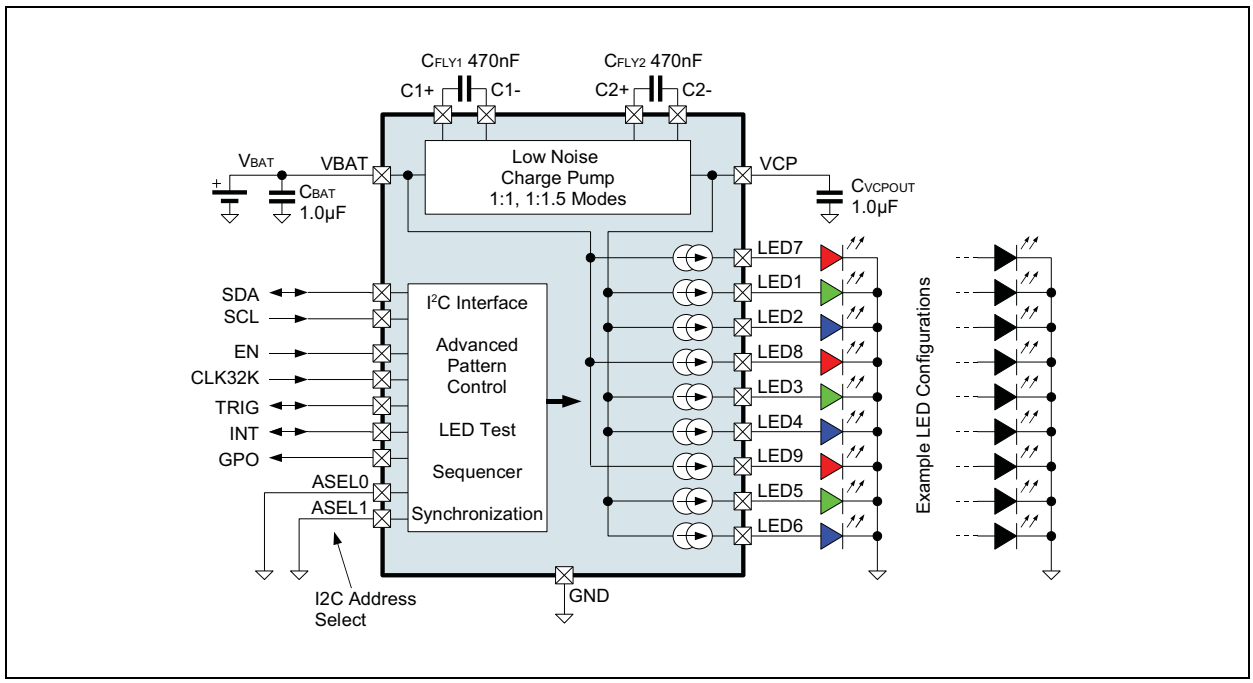
Applications

The AS3661, Programmable 9-channel LED Driver is ideal for fun and indicator lights, LED backlighting, and programmable current source.

Block Diagram

The functional blocks of this device for reference are shown below:

Figure 2:
AS3661 LED Driver Block Diagram



Pin Assignments

Figure 3:
Pin Out (Top View)

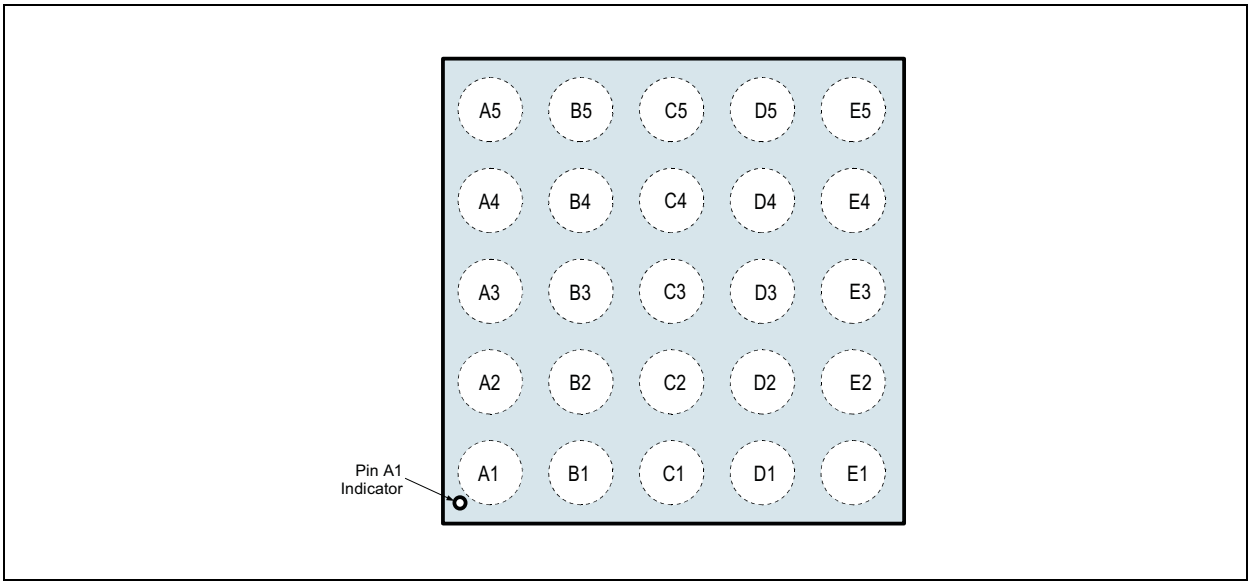


Figure 4:
Pin Description for AS3661

Pin Number	Pin Name	Description
A1	LED1	**LED1 Output**. Current source from V_{CP} .
A2	LED2	**LED2 Output**. Current source from V_{CP} .
A3	VCP	**Charge Pump output.** make a short connection to capacitor C_{VCPOUT}.
A4	C2-	**Charge Pump flying capacitor 2.** make a short connection to capacitor C_{FLY2}.
A5	C2+	**Charge Pump flying capacitor 2.** make a short connection to capacitor C_{FLY2}.
B1	LED3	**LED3 Output**. Current source from V_{CP} .
B2	LED4	**LED4 Output**. Current source from V_{CP} .
B3	ASEL1	**Digital input -I^2C address select**
B4	C1-	**Charge Pump flying capacitor 1.** make a short connection to capacitor C_{FLY1}.
B5	C1+	**Charge Pump flying capacitor 1.** make a short connection to capacitor C_{FLY1}.
C1	LED5	**LED5 Output**. Current source from V_{CP} .
C2	LED6	**LED6 Output**. Current source from V_{CP} .
C3	ASEL0	**Digital input -I^2C address select**
C4	EN	**Enable.** Active high digital input.

Pin Number	Pin Name	Description
C5	VBAT	**Positive Power Supply Input**
D1	LED7	**LED7 Output**. Current source from VBAT.
D2	LED8	**LED8 Output**. Current source from VBAT.
D3	INT	**Interrupt Output.** Open drain digital output for microcontroller unit, leave unconnected if not used.
D4	CLK32K	**Digital Clock Input.** Connect a 32kHz signal; if this signal is not available, connect this pin to GND.
D5	GND	**Ground**
E1	LED9	**LED9 Output**. Current source from VBAT.
E2	GPO	**General Purpose Output.** Leave unconnected if not used.
E3	TRIG	**Trigger Input.** Open drain, connect to ground if not used.
E4	SDA	**Serial-Data I/O**. Open drain digital I/O I^2C data pin.
E5	SCL	**Serial-Clock Input**

Absolute Maximum Ratings

Stresses beyond those listed in Absolute Maximum Ratings may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in Electrical Characteristics, is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Figure 5:
Absolute Maximum Ratings

Parameter	Min	Max	Units	Comments
VBAT, VCP, C1+, C1-, C2+, C2- to GND	-0.3	+7.0	V	
VCP to VBAT	-0.3		V	Diode between V_{CP} and VBAT
LED1, LED2 to LED9 to GND	-0.3	+7.0	V	
SDA, SCL, EN, CLK32K, TRIG, INT, GPO, ASEL0, ASEL1 to GND	-0.3	+7.0	V	
Electrostatic Discharge				
ESD HBM (LED1 to LED2)	8		kV	JEDEC JESD22-A114
ESD HBM (all other pins)	2.5		kV	
ESD MM	250		V	JEDEC JESD22-A115
ESD CDM	1		kV	JEDEC JESD22-C101
Temperature Ranges and Storage Conditions				
Continous Power Dissipation				Internally limited (overtemperature protection) (1)
Junction Temperature (T_{JMAX})		125	°C	
Storage Temperature Range	-55	125	°C	
Body Temperature during Soldering		260	°C	IPC/JEDEC J-STD-020
Junction to Ambient Thermal Resistance (θ_{JA}) (2)		87	°C/W	
Moisture Sensitivity Level	1			Represents a max. floor life time of *unlimited*
Recommended Operating Conditions				
Recommended charge pump load current	0	100	mA	

Note(s) and/or Footnote(s):

1. Internal thermal shutdown circuitry protects the device from permanent damage. Thermal shutdown engages at T_J = 150°C (typ.) and disengages at T_J = 130°C (typ.).
2. Junction to ambient thermal resistance is highly application and board-layout dependent. In applications where high maximum power dissipation exists, special care must be paid to thermal dissipation issues in board design.

Electrical Characteristics

V_{BAT} = 3.6V, V_{EN} = 1.65V, C_{BAT} = C_{VCPOUT} = 1.0µF, C_{FLY1-2} = 0.47µF, T_{AMB} = -30°C to 85°C, typical values @ T_{AMB} = 25°C (unless otherwise specified) (7).

Figure 6:
Electrical Characteristics

Symbol	Parameter	Condition	Min	Typ	Max	Unit
General Operating Conditions						
V_{BAT}	Supply Voltage		2.7		5.5	V
I_{VBAT}	Standby supply current	EN = 0V or CHIP_EN=0 (bit), external 32 kHz clock running or not running		1.4	4	µA
	Normal Mode supply current	External 32 kHz clock running, charge pump and current source outputs disabled		0.16	0.22	mA
		Charge pump in 1x mode, no load, current source outputs disabled		0.16	0.22	mA
		Charge pump in 1.5x mode, no load, current source outputs disabled		1.4		mA
	Power Save Mode supply current	External 32 kHz clock running		3.1	5	µA
		Internal oscillator running		0.16	0.23	mA
f_{OSC}	Internal Oscillator Frequency Accuracy	T_{AMB} = +25°C	-4		+4	%
			-7		+7	
T_{AMB}	Operating Temperature (1)		-30	25	85	°C
Charge Pump						
R_{OUT}	Charge Pump Output Resistance	Gain = 1.5 and V_{BAT} = 2.9V		6		Ω
		Gain = 1 and V_{BAT} = 2.9V		1		
		Gain = 1.5 and V_{BAT} = 3.6V		1.4		
		Gain = 1 and V_{BAT} = 3.6V		1		
f_{SW}	Switching Frequency		1.2	1.25	1.3	MHz
I_{GND}	Ground current	Gain = 1.5		1.2		mA
		Gain = 1		1		µA
t_{ON}	V_{CP} Turn-On Time (2)	V_{BAT} = 3.6V, I_{OUT} = 60 mA		100		µs
I_{LOAD}	Charge Pump load current	Recommended charge pump load current	0		100	mA

Symbol	Parameter	Condition		Min	Typ	Max	Unit
LED Driver							
I_{LEAK}	Leakage Current (LED1 to LED9)	PWM = 0%			0.1	1	µA
I_{MAX}	Maximum Source Current	Outputs LED1 to LED9			25.5		mA
I_{OUT}	Output Current [3] Accuracy	Output Current set to 17.5 mA	T_{AMB} = 25°C	-2.5%		+2.5%	%
				-5		+5	
I_{MATCH}	Matching	Output Current set to 17.5 mA			1	2.5	%
f_{LED}	LED Switching Frequency				312		Hz
V_{SAT}	Saturation Voltage [4]	Output Current set to 17.5 mA			45	100	mV
LED Test							
LSB	Least Significant Bit				30		mV
E_{ABS}	Total Unadjusted Error [5]	V_{IN_TEST} = 0V to V_{BAT}			<±3	±4	LSB
t_{CONV}	Conversion Time				2.7		ms
V_{IN_TEST}	DC Voltage Range			0		5	V
LOGIC INTERFACE							
Logic Input EN							
V_{IL}	Input Low Level					0.5	V
V_{IH}	Input High Level			1.2			V
I_{IN}	Input Current			-1.0		1.0	µA
t_{DELAY}	Input Delay [6]				2		µs
Logic Input SCL, SDA, TRIG, CLK32K, ASEL0, ASEL1							
V_{IL}	Input Low Level					0.2× V_{EN}	V
V_{IH}	Input High Level			0.8× V_{EN}			V
I_{IN}	Input Current			-1.0		1.0	µA
Logic Output SDA, TRIG, INT							
V_{OL}	Output Low Level	I_{OUT} = 3 mA (pull-up current)			0.3	0.5	V

Symbol	Parameter	Condition	Min	Typ	Max	Unit
I_L	Output Leakage Current	V_{CP} = 2.8V			1.0	µA
Logic Output GPO						
V_{OL}	Output Low Level	I_{OUT} = 3 mA		0.3	0.5	V
V_{OH}	Output High Level	I_{OUT} = -2 mA	V_{BAT} -0.5	V_{BAT} -0.3		V
I_L	Output Leakage Current	V_{CP} = 2.8V			1.0	µA
Logic Input CLK32K						
f_{CLK}	Clock Frequency			32.7		kHz
f_{CLKH}	High Time		6			µs
f_{CLKL}	Low Time		6			µs
t_R	Clock Rise Time	10 to 90%			2	µs
t_F	Clock Fall Time	90 to 10%			2	µs

Note(s) and/or Footnote(s):

1. In applications where high power dissipation and/or poor package thermal resistance is present, the maximum ambient temperature may have to be derated. Maximum ambient temperature ($T_{Amb-MAX}$) is dependent on the maximum operating junction temperature (T_{J-MAX} =125°C), the maximum power dissipation of the device in the application (P_{D-MAX}) and the junction to ambient thermal resistance of the part/package in the application (θ_{JA}) as given by the following equation: $T_{Amb-MAX} = T_{J-MAX} - (\theta_{JA} * P_{D-MAX})$.
2. Turn-on time is measured from the moment the charge pump is activated until the V_{CP} crosses 90% of its target value.
3. Output current accuracy is the difference between actual value of the output current and programmed value of this current. I_{MATCH} is determined as follows:
   For the constant current D1 to D9, the following are determined: The maximum current (max) and the minimum current (min), then the I_{MATCH} is calculated with: I_{MATCH} = 100*(((max-min)/2)+((max+min)/2))/((max+min)/2)- 100.
4. Saturation voltage is defined as the voltage when the LED current has dropped 10% from the value measured at V_{CP} - 1V.
5. Total unadjusted error includes offset, full-scale and linearity errors.
6. The I^2C host should allow at least 500µs before sending data the AS3661after the rising edge of the enable line.
7. Low-ESR Surface-Mount Ceramic Capacitors 8MLCCs) used in setting electrical characteristics.

Figure 7:
I²C Mode Timing Diagram

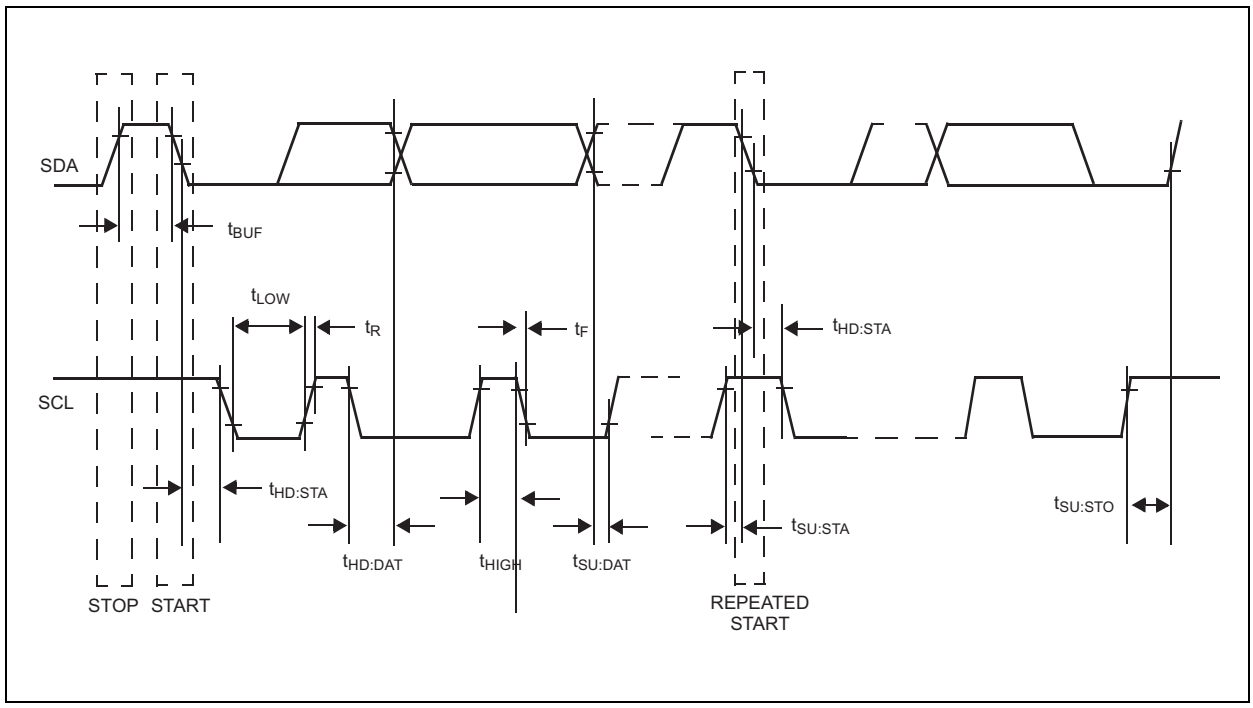


Figure 8:
Electrical Characteristics I²C(1)

Symbol	Parameter	Condition	Min	Typ	Max	Unit
I²C mode timings (see Figure 7)						
f_{SCLK}	SCL Clock Frequency		0		400	kHz
t_{BUF}	Bus Free Time Between a STOP and START Condition		1.3			µs
$t_{HD:STA}$	Hold Time (Repeated) START Condition (2)		0.6			µs
t_{LOW}	LOW Period of SCL Clock		1.3			µs
t_{HIGH}	HIGH Period of SCL Clock		0.6			µs
$t_{SU:STA}$	Setup Time for a Repeated START Condition		0.6			µs
$t_{HD:DAT}$	Data Hold Time (3)		50			ns
$t_{SU:DAT}$	Data Setup Time (4)		100			ns

Symbol	Parameter	Condition	Min	Typ	Max	Unit
t_R	Rise Time of Both SDA and SCL Signals		20 + 0.1C_B		300	ns
t_F	Fall Time of Both SDA and SCL Signals		15+ 0.1C_B		300	ns
$t_{SU:STO}$	Setup Time for STOP Condition		0.6			µs
C_B	Capacitive Load for Each Bus Line	Load of one picofarad corresponds to one nanosecond.	10		200	ns
$C_{I/O}$	I/O Capacitance (SDA, SCL)				10	pF

Note(s) and/or Footnote(s):

1. Specification is guaranteed by design and is not tested in production. V_{EN} = 1.65V to V_{BAT}.
2. After this period the first clock pulse is generated.
3. A device must internally provide a hold time of at least 300ns for the SDA signal (referred to the V_{IHMIN} of the SCL signal) to bridge the undefined region of the falling edge of SCL.
4. A fast-mode device can be used in a standard-mode system, but the requirement $t_{SU:DAT}$ = to 250ns must then be met. This is automatically the case if the device does not stretch the LOW period of the SCL signal. If such a device does stretch the LOW period of the SCL signal, it must output the next data bit to the SDA line t_R max + $t_{SU:DAT}$ = 1000 + 250 = 1250ns before the SCL line is released.

Typical Operating Characteristics

$V_{BAT} = 3.6V, V_{EN} = 1.65V, C_{BAT} = C_{VCPOUT} = 1.0\mu F, C_{FLY1-2} = 0.47\mu F,$ T_{AMB} = 25°C, unless otherwise specified.

Figure 9:
Charge Pump 1.5 × Efficiency vs. Load

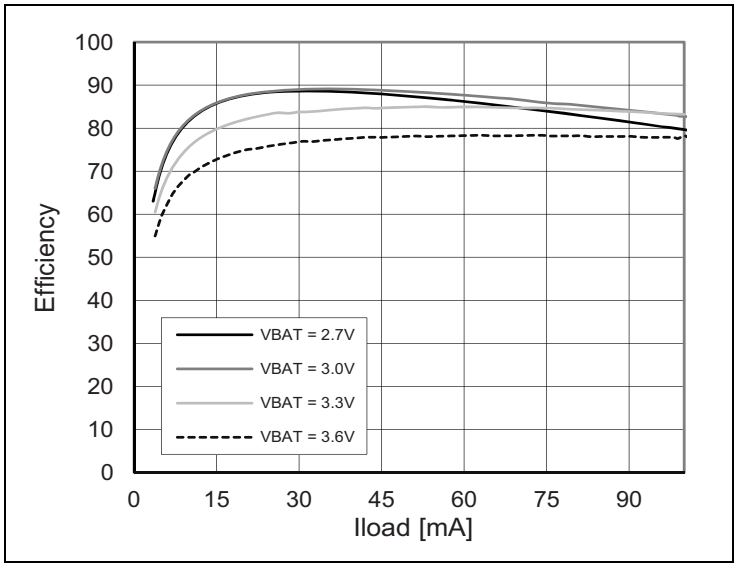


Figure 10:
Output Voltage vs. Load Current (1.5 × CP)

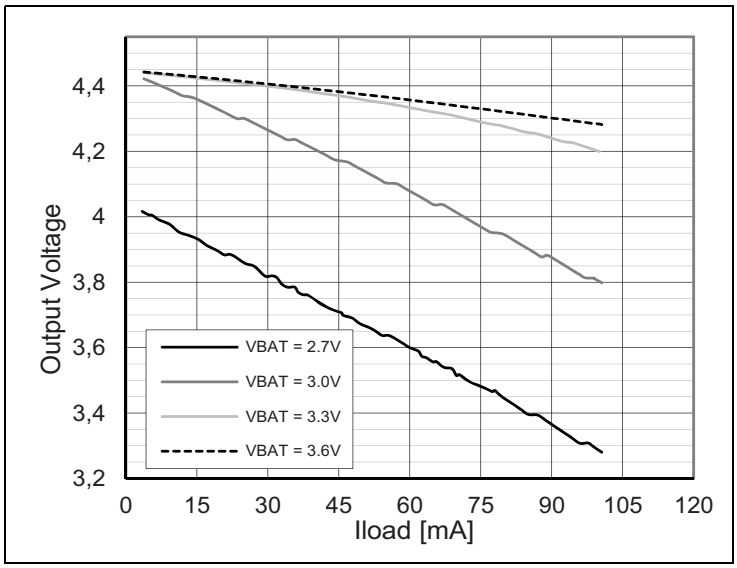


Figure 11:
Gain Change Hysteresis Loop (6 × 1mA load)

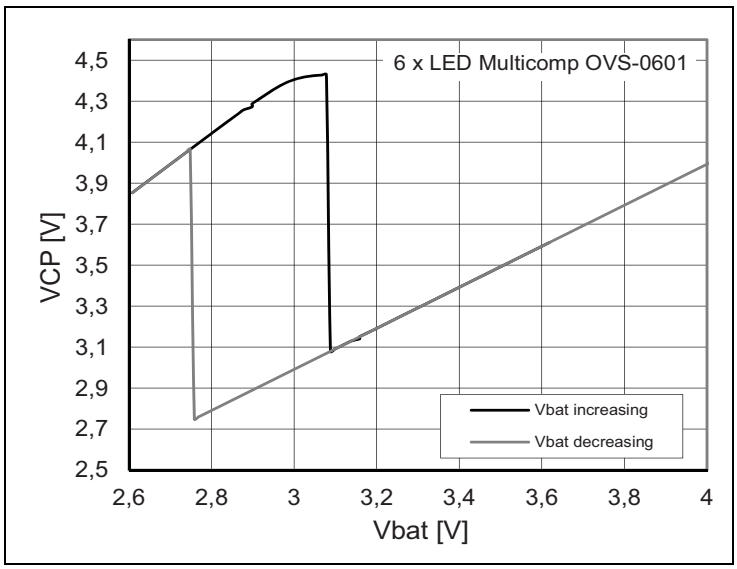


Figure 12:
Effect of Adaptive Hysteresis on Width of Hysteresis Loop

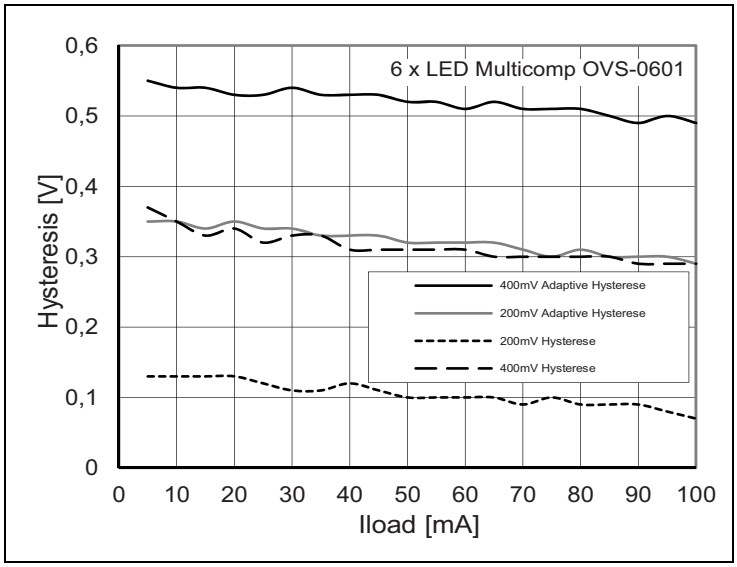


Figure 13:
LED Current Matching Distribution @17.5mA

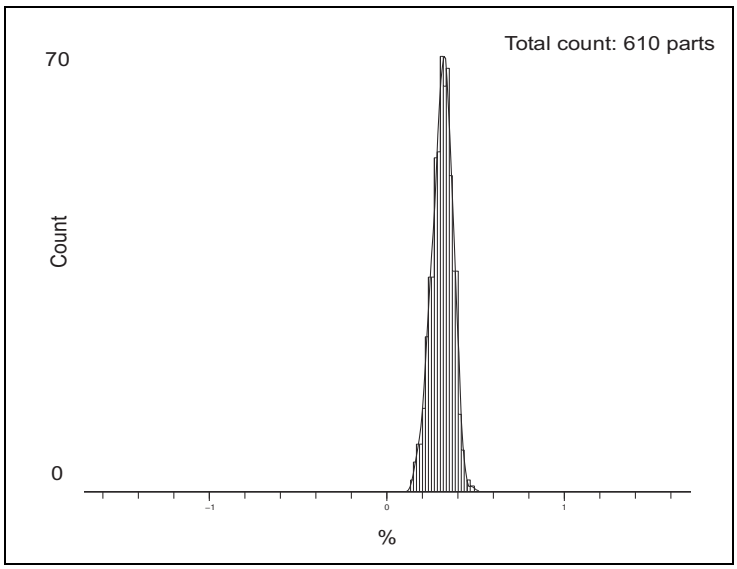


Figure 14:
LED Current Accuracy Distribution @17.5mA

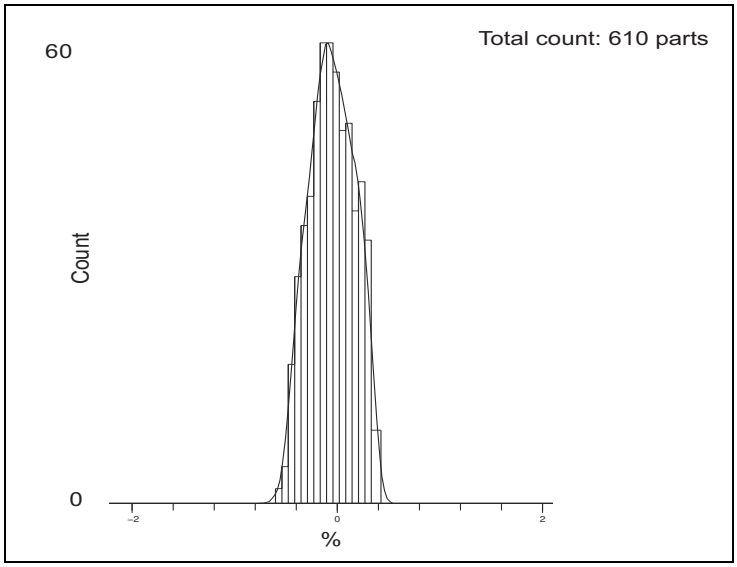


Figure 15:
Power Save Mode Supply Current vs.VBAT, Charge Pump in 1x Mode

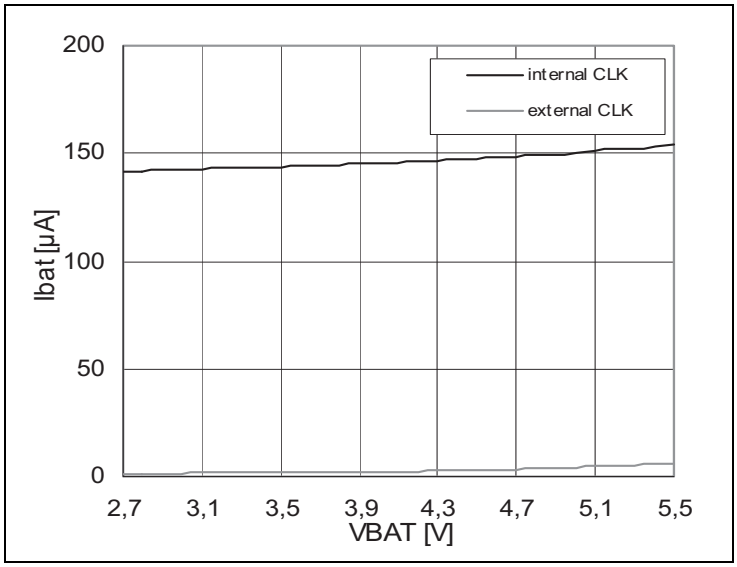


Figure 16:
Serial Bus Write and Charge Pump Startup, I_{LOAD} = 60mA

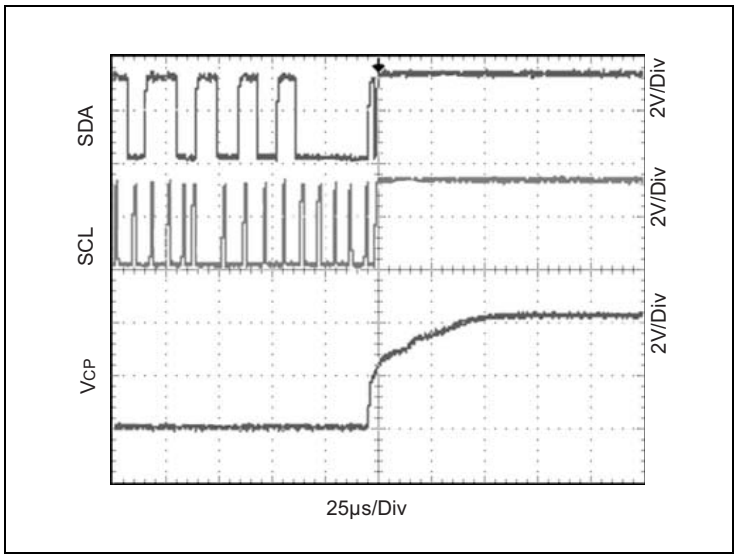


Figure 17:
Line Transient and Charge Pump Automatic Gain Change 1.5 to 1, 6LEDs@1mA 100% PWM

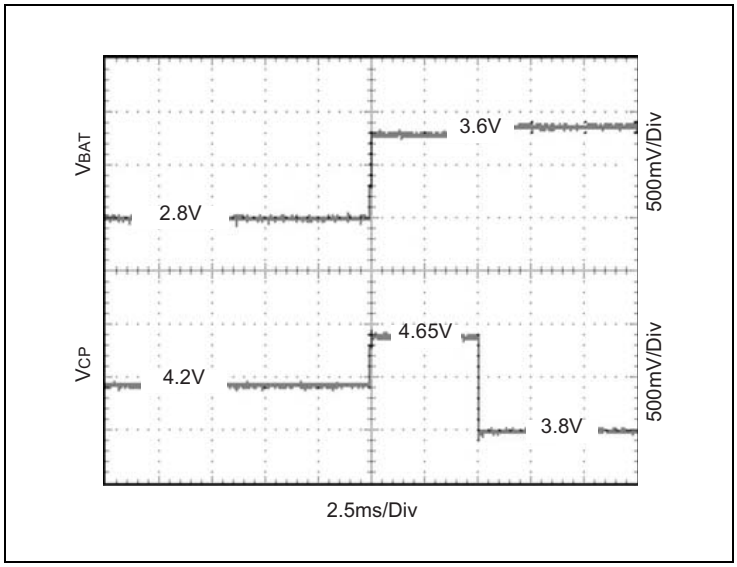


Figure 18:
Line Transient and Charge Pump Automatic Gain Change 1 to 1.5, 6LEDs@1mA 100% PWM

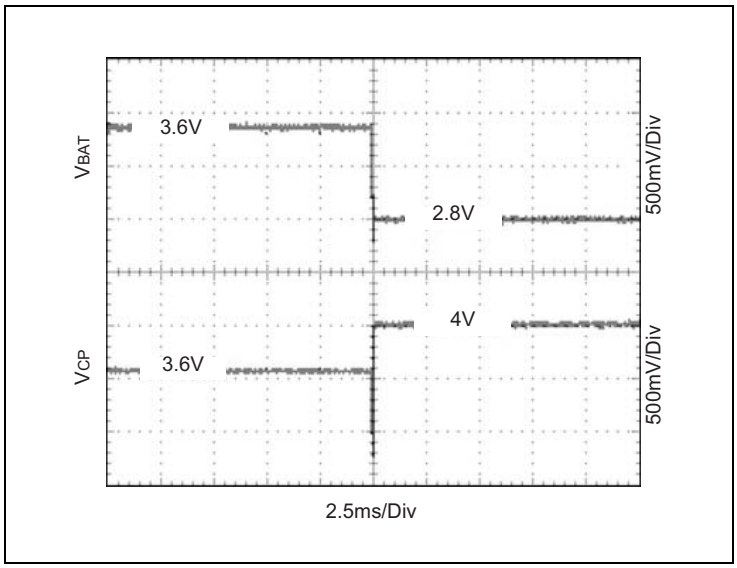


Figure 19:
100% PWM RGB LED Efficiency vs. VBAT

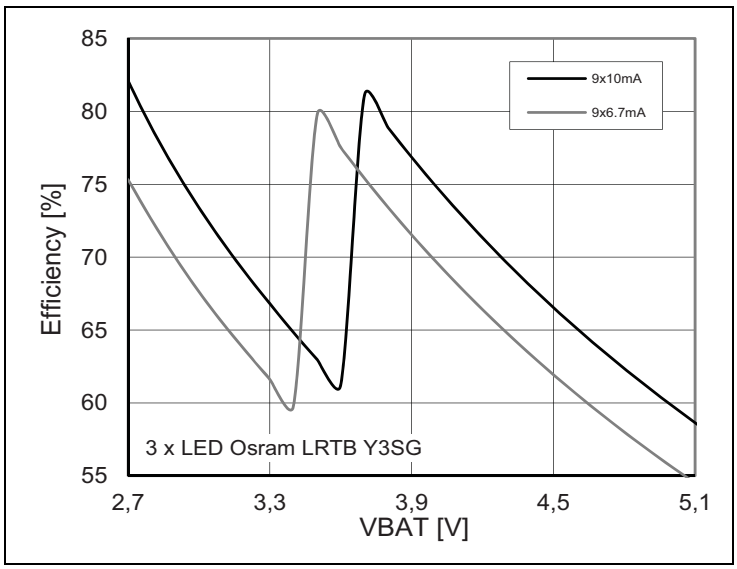
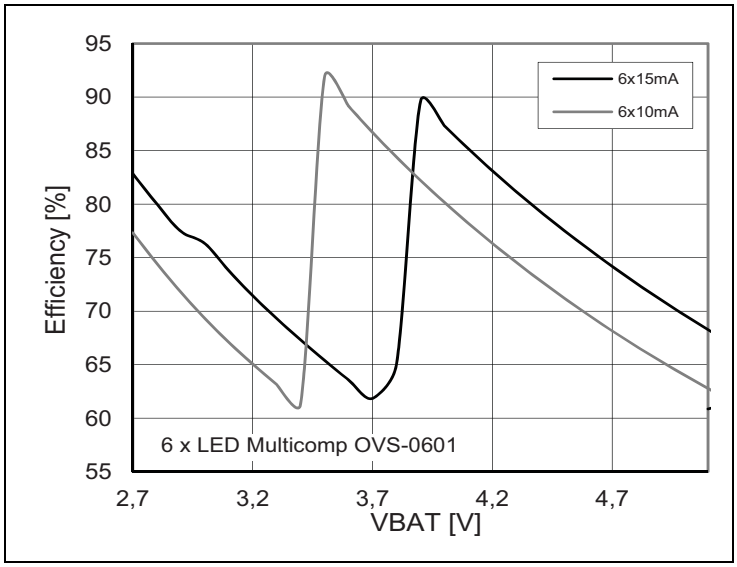


Figure 20:
100% PWM WLED Efficiency vs. VBAT



Detailed Description

The AS3661 is a fully integrated lighting management unit for producing lighting effects for mobile devices. The AS3661 includes all necessary power management, high-side current sources, temperature compensation, two wire control interface and programmable pattern generators. The overall maximum current for each driver is set by an 8-bit register. The AS3661 controls LED luminance with a pulse width modulation (PWM) scheme with a resolution of 12 bits. The temperature compensation is also done by a PWM.

Programming

The AS3661 provides flexibility and programmability for dimming and sequencing control. Each LED can be controlled directly and independently through the serial bus or LED drivers can be grouped together for pre-programmed flashing patterns. The AS3661 has three independent program execution engines, so it is possible to form three independently programmable LED banks. LED drivers can be grouped based on their function so that, for example, the first bank of drivers can be assigned to the keypad illumination, the second bank to the “funlights” and the third group to the indicator LED(s). Each bank can contain 1 to 9 LED driver outputs. Instructions for program execution engines are stored in the program memory. The total amount of the program memory is 96 instructions and the user can allocate the memory as required by the engines.

LED Error Detection

AS3661 has a built-in LED error detection. Error detection does not only detect open and short circuit, but provides an opportunity to measure the V_F’s of the LEDs. The test event is activated by a serial interface write and the result can be read through the serial interface during the next cycle. This feature can also be addressed to measure the voltage on VBAT, VCP and INT pins. Typical example usage includes monitoring battery voltage or using INT pin as a light sensor interface.

Energy Efficiency

When charge pump automatic mode selection is enabled, the AS3661 monitors the voltage over the drivers of LED1 to LED6 so that the device can select the best charge pump gain and maintain good efficiency over the whole operating voltage range. The red LED element of an RGB LED typically has a forward voltage of about 2V. For that reason, the outputs LED7, LED8 and LED9 are internally powered by VBAT, since battery voltage is high enough to drive red LEDs over the whole operating voltage range. This allows to drive three RGB LEDs with good efficiency because the red LEDs doesn't load the charge pump. AS3661 is able to automatically enter power-save mode, when LED outputs are not active and thus lowering idle current consumption down to 10µA (typ.). During the

"downtime" of the PWM cycle (constant current output status is low) additional power savings can be achieved when the PWM power save feature is enabled.

Temperature Compensation

The luminance of an LED is typically a function of its temperature even though the current flowing through the LED remains constant. Since luminance is temperature dependent, many LED applications require some form of temperature compensation to decrease luminance and color purity variations due to temperature changes. The AS3661 has a build in temperature sensing element and PWM duty cycle of the LED drivers changes linearly in relationship to changes in temperature. User can select the slope of the graph (31 slopes) based on the LED characteristics. This compensation can be done either constantly, or only right after when the device wakes up from power save mode, to avoid error due to self-heating of the device. Linear compensation is considered to be practical and accurate enough for most LED applications. Compensation is effective over the temperature range from -40°C to 90°C.

Figure 21:
Temperature Compensation Principle

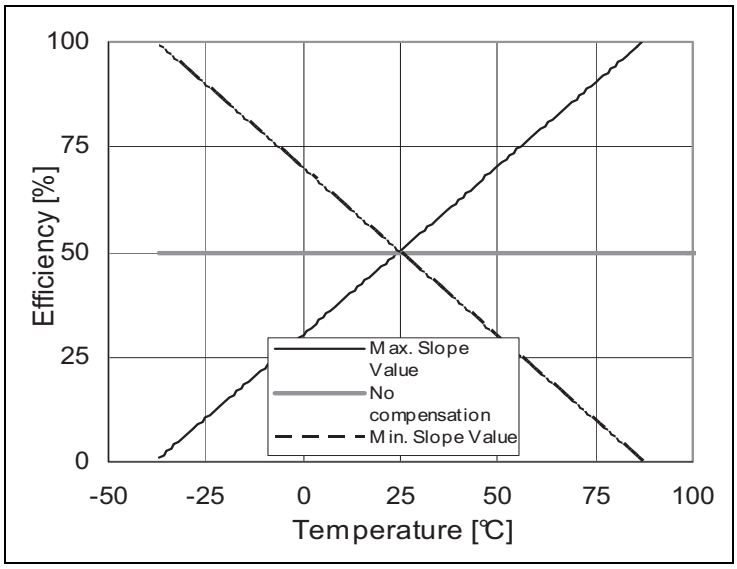
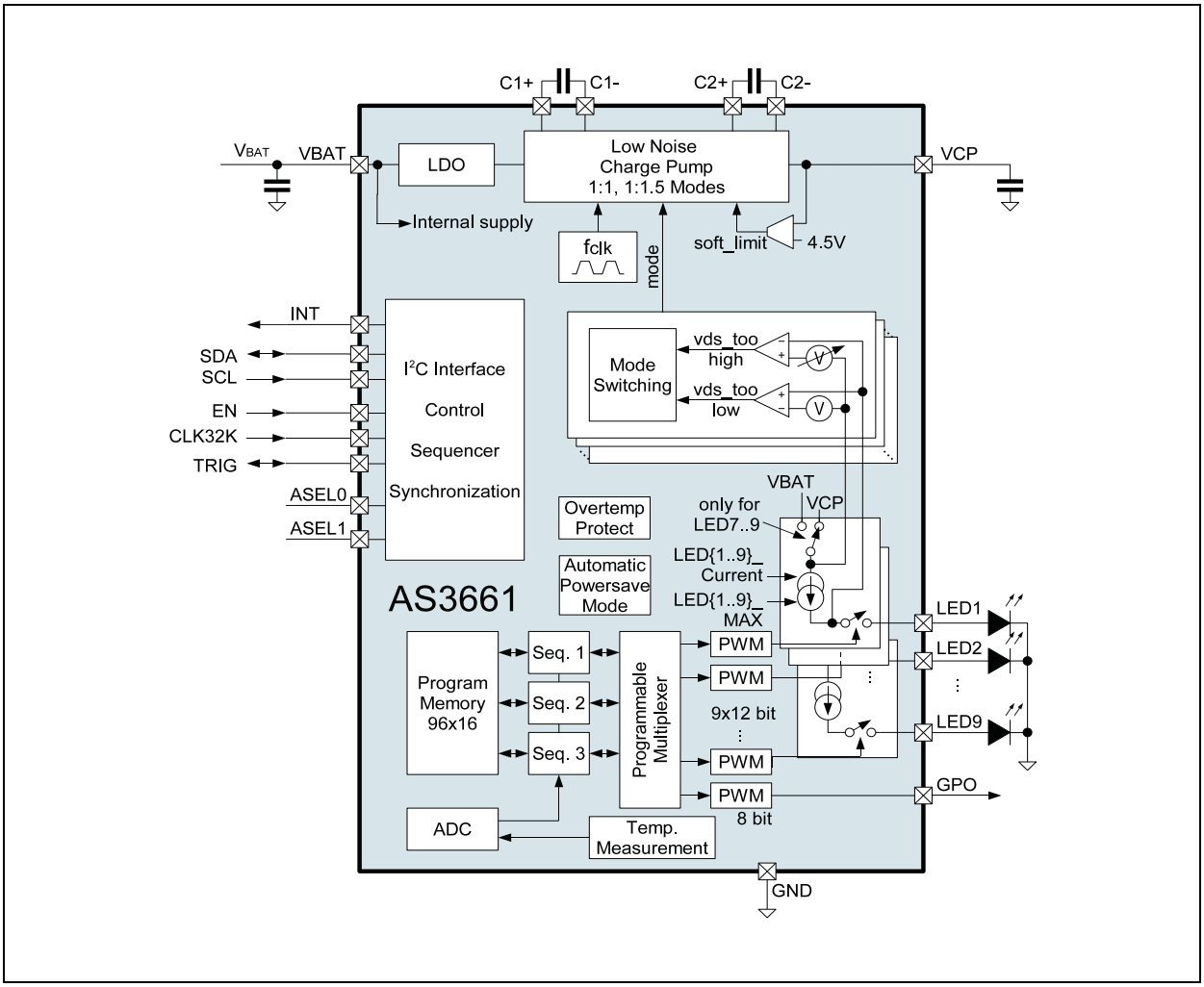


Figure 22:
AS3661 - Block Diagram



Modes of Operation

The following are the different modes of operation of AS3661

RESET

In the RESET mode all the internal registers are reset to the default values. Reset is entered always if Reset Register (3DH) is written FFH or internal Power ON Reset is active. Power ON Reset (POR) will activate during the chip startup or when the supply voltage V_{BAT} fall below 1.5V (typ.). Once V_{BAT} rises above 1.5V (typ.) POR will be inactivate and the chip will continue to the STANDBY mode. CHIP_EN control bit is low after POR by default.

STANDBY

The STANDBY mode is entered if the register bit CHIP_EN or EN pin is logic low and Reset is not active. This is the low power consumption mode, when all circuit functions are disabled. Registers can be written in this mode if EN pin is logic high so that the control bits will be effective right after the start up.

STARTUP

When CHIP_EN bit is written high and the EN pin is high, the INTERNAL STARTUP SEQUENCE powers up all the needed internal blocks (V_{REF}, Bias, Oscillator etc.). Startup delay is 500 µs. If the chip temperature rises too high, the Thermal Shutdown (TSD) disables the chip operation and chip waits in STARTUP mode until no thermal shutdown event is present.

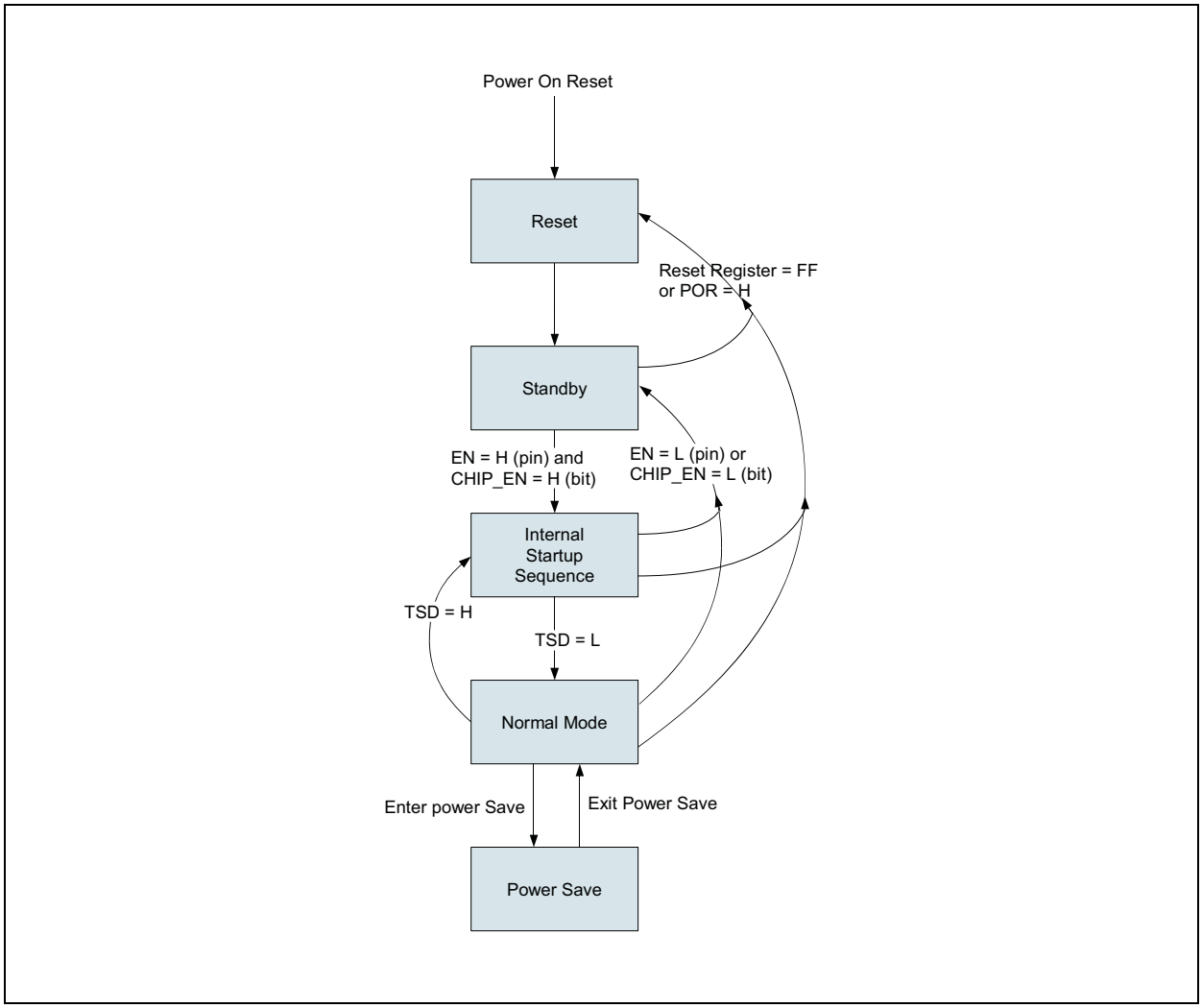
NORMAL

During NORMAL mode the user controls the chip using the Control Registers.

POWER SAVE

In POWER SAVE mode analog blocks are disabled to minimize power consumption. (see Automatic Power Save Mode).

Figure 23:
Mode Select

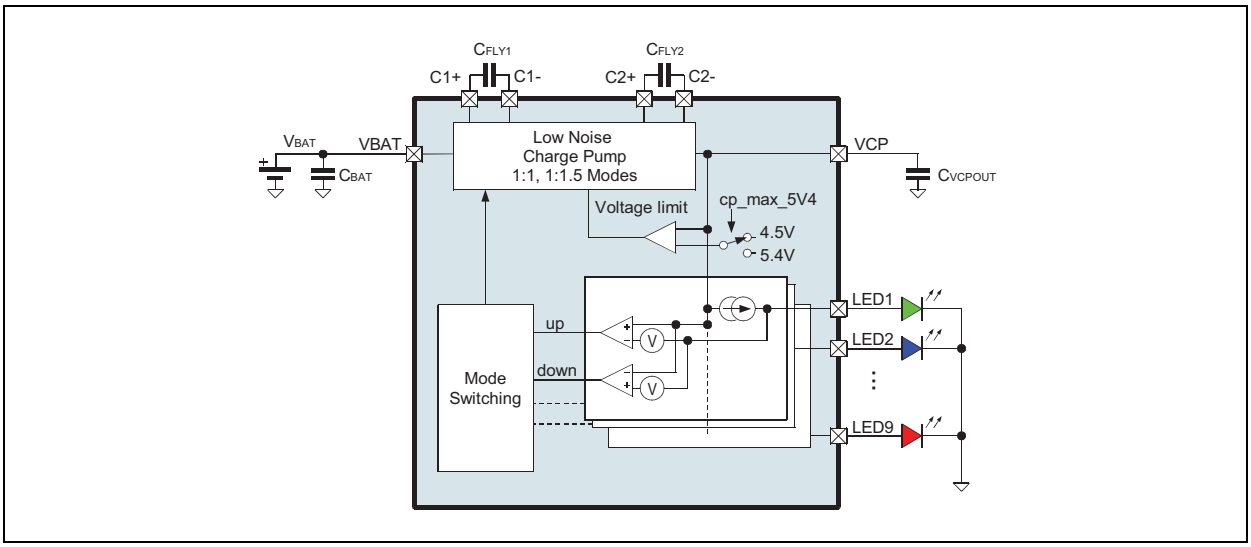


Charge Pump Operational Description

Overview

The AS3661 includes a pre-regulated switched-capacitor charge pump with a programmable voltage multiplication of 1 and 1.5x. In 1.5x mode by combining the principles of a switched-capacitor charge pump and a linear regulator, it generates a regulated 4.5V output from Li-Ion input voltage range. A two-phase non-overlapping clock generated internally controls the operation of the charge pump. During the charge phase, both flying capacitors (C_{FLY1} and C_{FLY2}) are charged from input voltage. In the pump phase that follows, the flying capacitors are discharged to output. A traditional switched capacitor charge pump operating in this manner will use switches with very low on-resistance, ideally 0Ω, to generate an output voltage that is 1.5x the input voltage. The AS3661 regulates the output voltage by controlling the resistance of the input-connected pass-transistor switches in the charge pump.

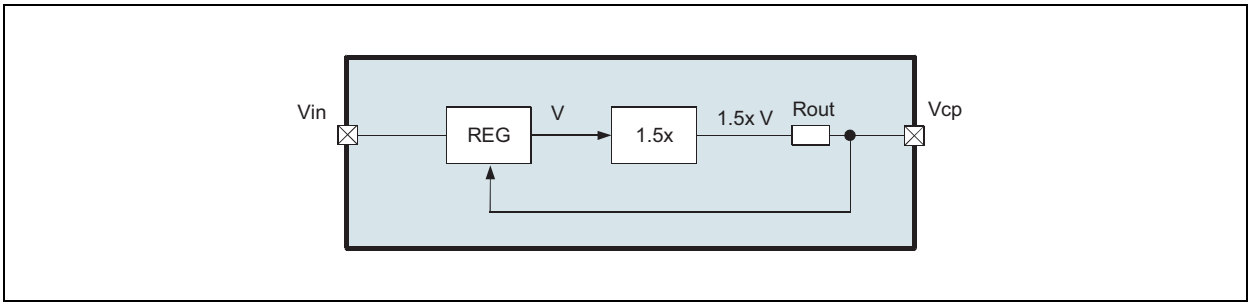
Figure 24:
Charge Pump



Output Resistance

At lower input voltages, the charge pump output voltage may degrade due to effective output resistance (R_{OUT}) of the charge pump. The expected voltage drop can be calculated by using a simple model for the charge pump illustrated in Figure 25 below.

Figure 25:
Charge Pump Output Resistance



The model shows a linear pre-regulation block (REG), a voltage multiplier (1.5x), and an output resistance (R_{OUT}). The output resistance models the output voltage drop that is inherent to switched capacitor converters. The output resistance is 3.5Ω (typ.), and it is a function of switching frequency, input voltage, flying capacitors' capacitance value, internal resistances of the switches and ESR of the flying capacitors. When the output voltage is in regulation, the regulator in the model controls the voltage V to keep the output voltage equal to 4.5V (typ.). With increased output current, the voltage drop across R_{OUT} increases. To prevent drop in output voltage, the voltage drop across the regulator is reduced, V increases, and V_{CP} remains at 4.5V. When the output current increases to the point that there is zero voltage drop across the regulator, V equals the input voltage, and the output voltage is "on the edge" of regulation. Additional output current causes the output voltage to fall out of regulation, so that the operation is similar to a basic open-loop 1.5x charge pump. In this mode, output current results in output voltage drop proportional to the output resistance of the charge pump. The out-of-regulation output voltage can be approximated by:

(EQ1) $$V_{CP} = 1.5 \times V_{IN} - I_{OUT} \times R_{OUT}.$$

Controlling the Charge Pump

The charge pump is controlled with two CP_MODE bits in MISC register (address 36H). When both of the bits are low, the charge pump is disabled and the output voltage is pulled down with an internal 300 kΩ (typ.) resistor. The charge pump can be forced to bypass mode, so that the battery voltage is connected directly to the current sources. In 1.5x mode the output voltage is boosted to 4.5V. In automatic mode the charge pump operation mode is determined by saturation of constant current drivers, like described in chapter LED Forward Voltage Monitoring.

LED Forward Voltage Monitoring

When the charge pump automatic mode selection is enabled, the voltages over the LED drivers LED1 to LED6 are monitored.

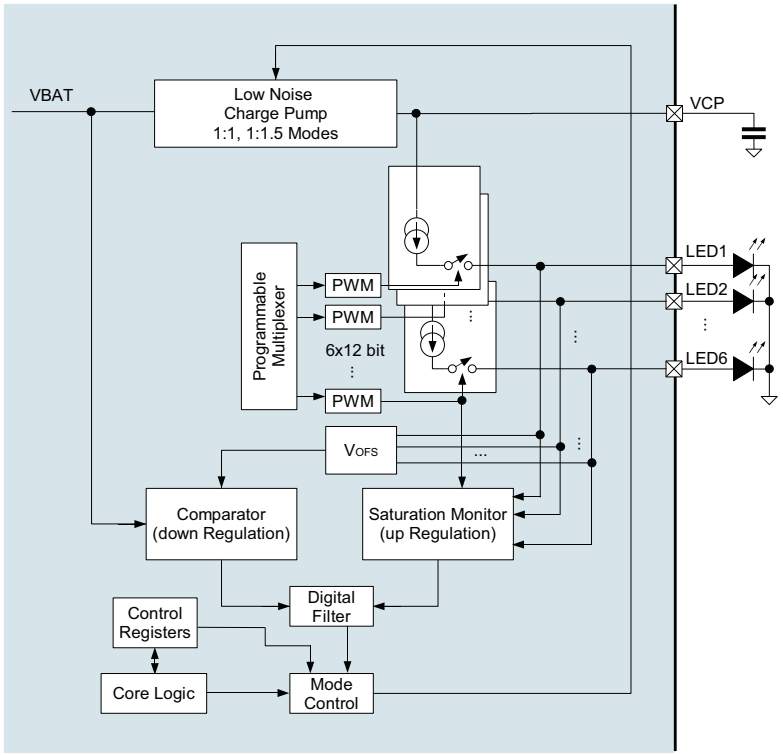
Note(s): Power input for current source outputs LED7, LED8 and LED9 are internally connected to the VBAT pin.

If the LED1 to LED6 drivers do not have enough headroom, the charge pump gain is set to 1.5x. Driver saturation monitor does not have a fixed voltage limit, since saturation voltage is a function of temperature and current. The charge pump gain is set to 1x, when the battery voltage is high enough to supply all LEDs. In automatic gain change mode, the charge pump is switched to bypass mode (1x), when LEDs are inactive for over 50 ms.

Gain Change Hysteresis

The charge pump gain control utilizes digital filtering to prevent supply voltage disturbances (for example, the transient voltage on the power supply during the GSM burst) from triggering unnecessary gain changes. Hysteresis is provided to prevent periodic gain changes, which would occur due to LED driver and charge pump voltage drop in 1x mode. The hysteresis of the gain change is user configurable, default setting is factory programmable. Flexible configuration ensures, that the hysteresis can be minimized or set to desired level in each application. LED forward voltage monitoring and gain control block diagram is shown in Figure 26.

Figure 26:
Forward Voltage Monitoring and Gain Control Block



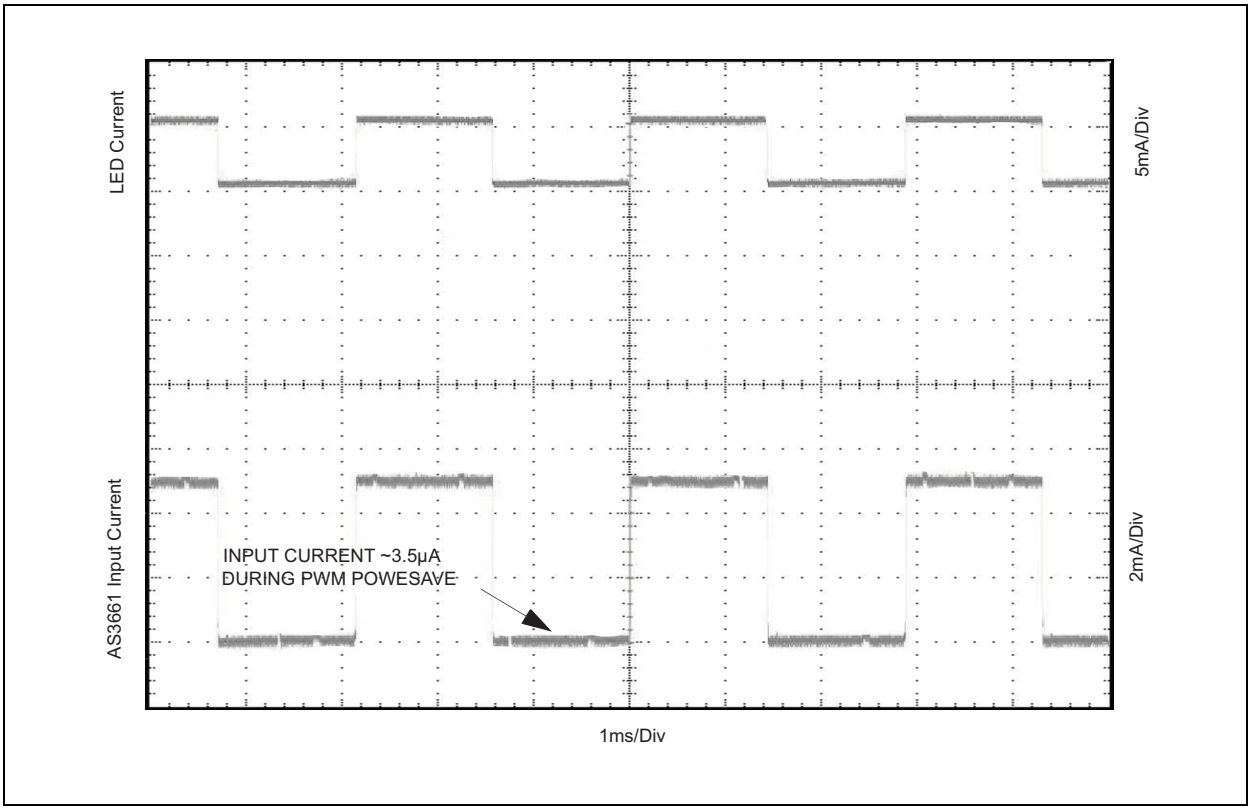
Automatic Power Save Mode

Automatic power save mode is enabled when POWERSAVE_EN bit in register address 36H is ‘1’. Almost all analog blocks are powered down in power save, if an external clock signal is used. Only the charge pump protection circuits remain active. However, if the internal clock has been selected, only charge pump and LED drivers are disabled during the power save; the digital part of the LED controller needs to stay active. In both cases the charge pump enters to the weak 1x mode. In this mode the charge pump utilizes a passive current limited keep-alive switch, which keeps the output voltage at the battery level. During the program execution AS3661 can enter power save if there is no PWM activity in any of the LED driver outputs. To prevent short power save sequences during program execution, AS3661 has an instruction look-ahead filter. During program execution engine 1, engine 2 and engine 3 instructions are constantly analyzed, and if there is time intervals of more than 50ms in length with no PWM activity on LED driver outputs, the device will enter power save. In power save mode program execution continues uninterruptedly. When an instruction that requires PWM activity is executed, a fast internal startup sequence will be started automatically.

PWM Power Save Mode

PWM cycle power save mode is enabled when register 36 bit [2] PWM_PS_EN is set to '1'. In PWM power save mode analog blocks are powered down during the "down time" of the PWM cycle. Blocks that are powered down depends whether external or internal clock is used. While the Automatic Power Save Mode (see above) saves energy when there is no PWM activity at all, the PWM Power Save mode saves energy during PWM cycles. Like the Automatic Power Save Mode, PWM Power Save Mode works also during program execution.

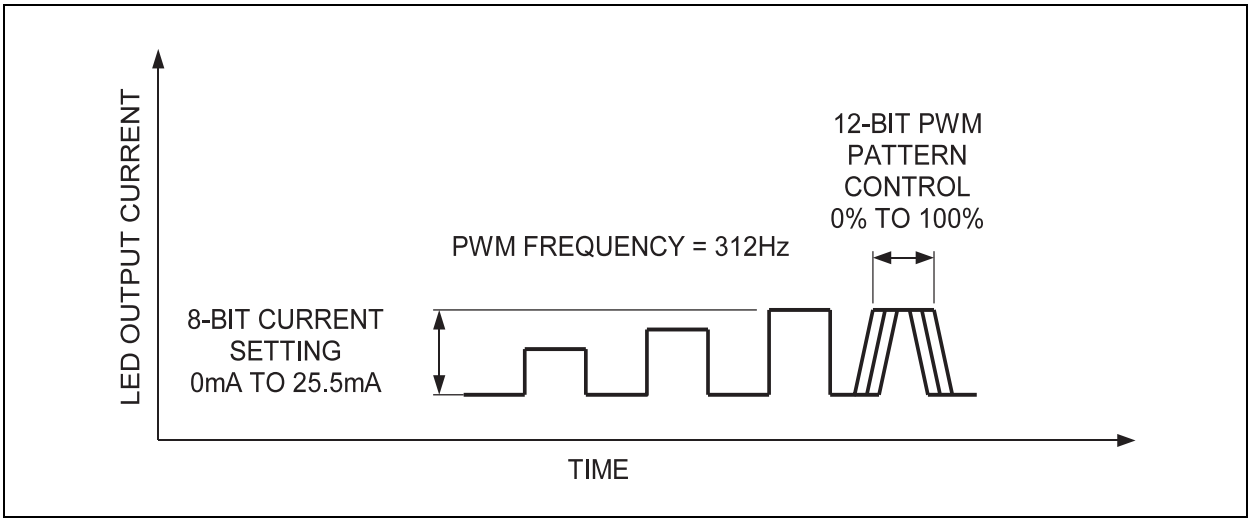
Figure 27:
PWM Powersave Principle with External Clock (VDD =3.6V, 50% PWM, I_{LED9}=5mA)



LED Driver Operational Description

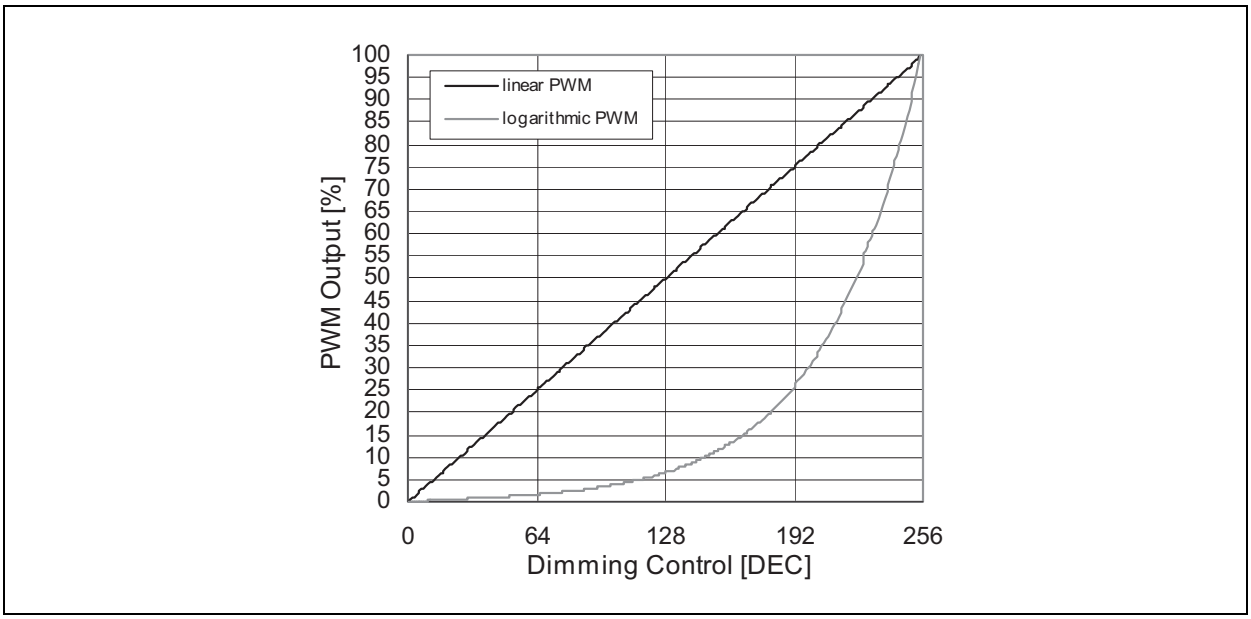
AS3661 LED drivers are constant current sources. The output current can be programmed by control registers up to 25.5 mA. The overall maximum current is set by 8-bit output current control registers with 100 µA step size. Each of the 9 LED drivers has a separate output current control register. The LED luminance pattern (dimming) is controlled with PWM (pulse width modulation) technique, which has internal resolution of 12 bits (8-bit control can be seen by user). PWM frequency is 312 Hz (see Figure 28).

Figure 28:
LED Pattern and Current Control Principle



LED dimming is controlled according to a logarithmic or linear scale (see Figure 29). Logarithmic or linear scheme can be set for both the program execution engine control and direct PWM control.

Figure 29:
Logarithmic vs. Linear Dimming



Note(s): If the temperature compensation is active, the maximum PWM duty cycle is limited to 50% at 25°C. This is required to allow enough headroom for temperature compensation over the whole temperature range -40 °C to 90°C.

Powering LEDs

Although the AS3661 is very suitable for white LED and general purpose applications, it is particularly well suited to use with RGB LEDs. The AS3661 architecture is optimized for use with three RGB LEDs. Typically, the red LEDs have forward voltages below 2V and thus red LEDs can be powered directly from V_{BAT}. In AS3661 the LED7, LED8 and LED9 drivers are directly powered from the battery voltage (V_{BAT}), not from the charge pump output. The LED1 to LED6 drivers are internally connected to the charge pump output and these outputs can be used for driving green and blue (V_F = 2.7V to 3.7V) or white LEDs. Of course, LED7, LED8 and LED9 outputs can be used for green, blue or white LEDs if the VBAT voltage is high enough. An RGB LED configuration example is given in the Typical Applications section.

Controlling the High-side LED Drivers

- Direct PWM Control

  All AS3661 LED drivers, LED1 to LED9, can be controlled independently through the two-wire serial I^2C compatible interface. For each high-side driver there is a PWM control register. Direct PWM control is active by default.

- Controlling by Program Execution Engines

  Engine control is used when the user wants to create programmed sequences. The program execution engine has higher priority than direct control registers. Therefore if the user has set to PWM register a certain value it will be automatically overridden when the program execution engine controls the driver. LED control and program execution engine operation is described in the chapter Control Register Details.

- Master Fader Control

  In addition to LED-by-LED PWM register control, the AS3661 is equipped with so called master fader control, which allows the user to fade in or fade out multiple LEDs by writing to only one register. This is an useful function to minimize serial bus traffic between the MCU and the AS3661. The AS3661 has three master fader registers, so it is possible to form three master fader groups. Master fader control can be used with the engines as well.

I²C Compatible Control Interface

The AS3661 supports the I²C bus protocol. A device that sends data onto the bus is defined as a transmitter and a device receiving data as a receiver. The device that controls the message is called a master. The devices that are controlled by the master are referred to as slaves. A master device that generates the serial clock (SCL), controls the bus access, and generates the START and STOP conditions must control the bus.

The AS3661 operates as a slave on the I²C bus. Within the bus specifications a standard mode (100kHz maximum clock rate) and a fast mode (400kHz maximum clock rate) are defined. The AS3661 works in both modes. Connections to the bus are made through the open-drain I/O lines SDA and SCL Figure 30.

I²C Address Selection

The slave address can be selected depending on the connection of the two address selection pins ASEL0 and ASEL1. The selected address for reading and writing depending on the state of ASEL0 and ASEL1 can be found in Figure 30 below.

Figure 30:
Chip Address Configuration

ASEL1	ASEL0	Address (Hex)	8 bit Hex Address R/W
GND	GND	32	64/65
GND	V_{EN}	33	66/67
V_{EN}	GND	34	68/69
V_{EN}	V_{EN}	35	6A/6B

The following bus protocol has been defined (Figure 31):

- Data transfer may be initiated only when the bus is not busy.
- During data transfer, the data line must remain stable whenever the clock line is HIGH. Changes in the data line while the clock line is HIGH are interpreted as control signals.

Accordingly, the following bus conditions have been defined:

Bus Not Busy

Both data and clock lines remain HIGH.

Start Data Transfer

A change in the state of the data line, from HIGH to LOW, while the clock is HIGH, defines a START condition.

Stop Data Transfer

A change in the state of the data line, from LOW to HIGH, while the clock line is HIGH, defines the STOP condition.

Data Valid

The state of the data line represents valid data when, after a START condition, the data line is stable for the duration of the HIGH period of the clock signal. The data on the line must be changed during the LOW period of the clock signal. There is one clock pulse per bit of data.

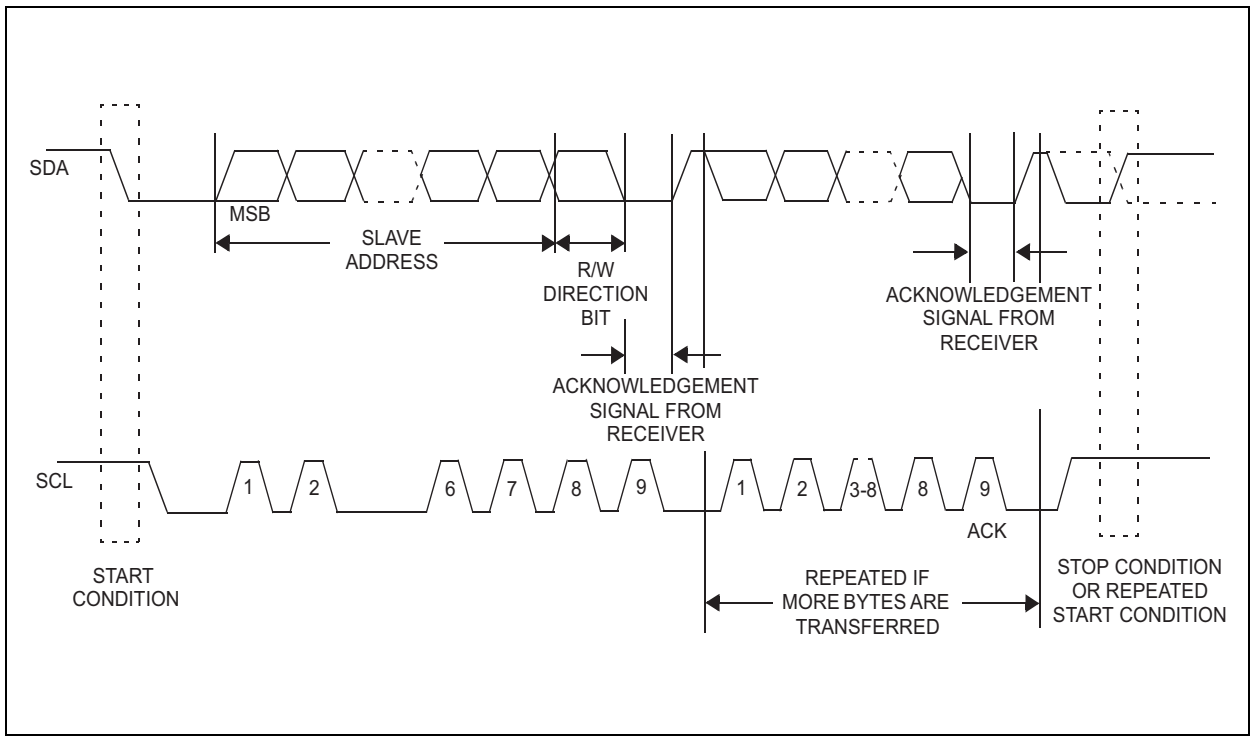
Each data transfer is initiated with a START condition and terminated with a STOP condition. The number of data bytes transferred between START and STOP conditions are not limited, and are determined by the master device. The information is transferred byte-wise and each receiver acknowledges with a ninth bit.

Acknowledge

Each receiving device, when addressed, is obliged to generate an acknowledge after the reception of each byte. The master device must generate an extra clock pulse that is associated with this acknowledge bit.

A device that acknowledges must pull down the SDA line during the acknowledge clock pulse in such a way that the SDA line is stable LOW during the HIGH period of the acknowledge-related clock pulse. Of course, setup and hold times must be taken into account. A master must signal an end of data to the slave by not generating an acknowledge bit on the last byte that has been clocked out of the slave. In this case, the slave must leave the data line HIGH to enable the master to generate the STOP condition.

Figure 31:
Data Transfer on I²C Serial Bus



Depending upon the state of the R/W bit, two types of data transfer are possible:

1. **Data transfer from a master transmitter to a slave receiver.** The first byte transmitted by the master is the slave address. Next follows a number of data bytes. The slave returns an acknowledge bit after each received byte. Data is transferred with the most significant bit (MSB) first.
2. **Data transfer from a slave transmitter to a master receiver.** The master transmits the first byte (the slave address). The slave then returns an acknowledge bit, followed by the slave transmitting a number of data bytes. The master returns an acknowledge bit after all received bytes other than the last byte. At the end of the last received byte, a "not acknowledge" is returned. The master device generates all of the serial clock pulses and the START and STOP conditions. A transfer is ended with a STOP condition or with a repeated START condition. Since a repeated START condition is also the beginning of the next serial transfer, the bus is not released. Data is transferred with the most significant bit (MSB) first.

The AS3661 can operate in the following two modes:

1. **Slave Receiver Mode (Write Mode):** Serial data and clock are received through SDA and SCL. After each byte is received an acknowledge bit is transmitted. START and STOP conditions are recognized as the beginning and end of a serial transfer. Address recognition is performed by hardware after reception of the slave address and direction bit (see Figure 32). The slave

address byte is the first byte received after the master generates the START condition. The slave address byte contains the 7-bit AS3661 address, which is 0110010[1], followed by the direction bit (R/W), which, for a write, is 0[2]. After receiving and decoding the slave address byte the device outputs an acknowledge on the SDA line. After the AS3661 acknowledges the slave address + write bit, the master transmits a register address to the AS3661. This sets the register pointer on the AS3661. The master may then transmit zero or more bytes of data (if more than one data byte is written see also Blockwrite/read boundaries), with the AS3661 acknowledging each byte received. The address pointer will increment after each data byte is transferred. The master generates a STOP condition to terminate the data write.

2. **Slave Transmitter Mode (Read Mode):** The first byte is received and handled as in the slave receiver mode. However, in this mode, the direction bit indicates that the transfer direction is reversed. Serial data is transmitted on SDA by the AS3661 while the serial clock is input on SCL. START and STOP conditions are recognized as the beginning and end of a serial transfer (Figure 32 and Figure 33). The slave address byte is the first byte received after the master generates a START condition. The slave address byte contains the 7-bit AS3661 address, which is 0110010, followed by the direction bit (R/W), which, for a read, is 1[3]. After receiving and decoding the slave address byte the device outputs an acknowledge on the SDA line. The AS3661 then begins to transmit data starting with the register address pointed to by the register pointer (if more than one data byte is read see Blockwrite/read boundaries). If the register pointer is not written to before the initiation of a read mode the first address that is read is the last one stored in the register pointer. The AS3661 must receive a “not acknowledge” to end a read.

1. ‘The I2C address’ depends on the external connection of ASEL0 and ASEL1; see Chip Address Configuration.
2. The address for writing to the AS3661 is 8Xh = 01100100b - see Figure 30.
3. The address for read mode from the AS3661 is 8Xh+1 = 01100101b - Figure 30.

Figure 32:
Data Write - Slave Receiver Mode

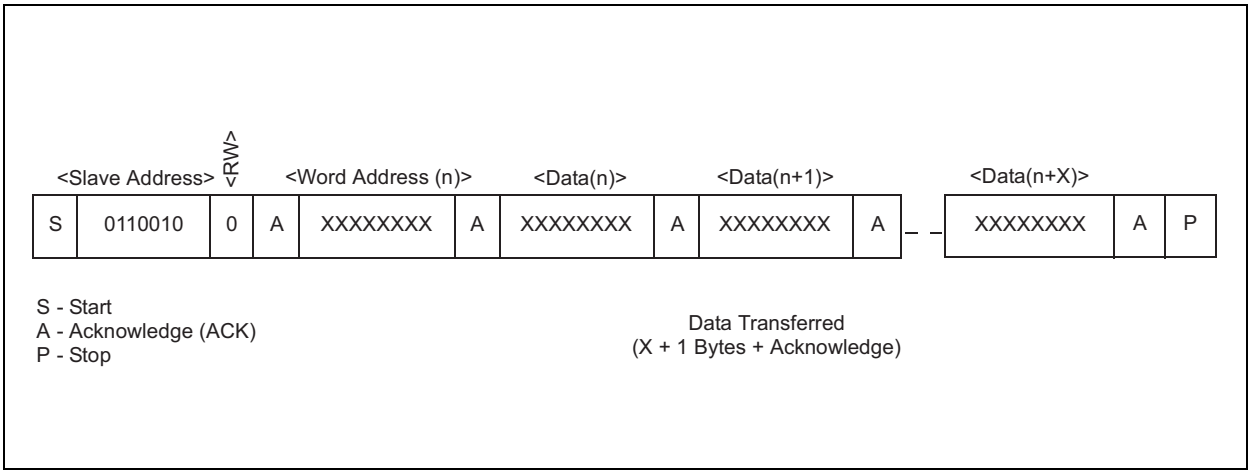
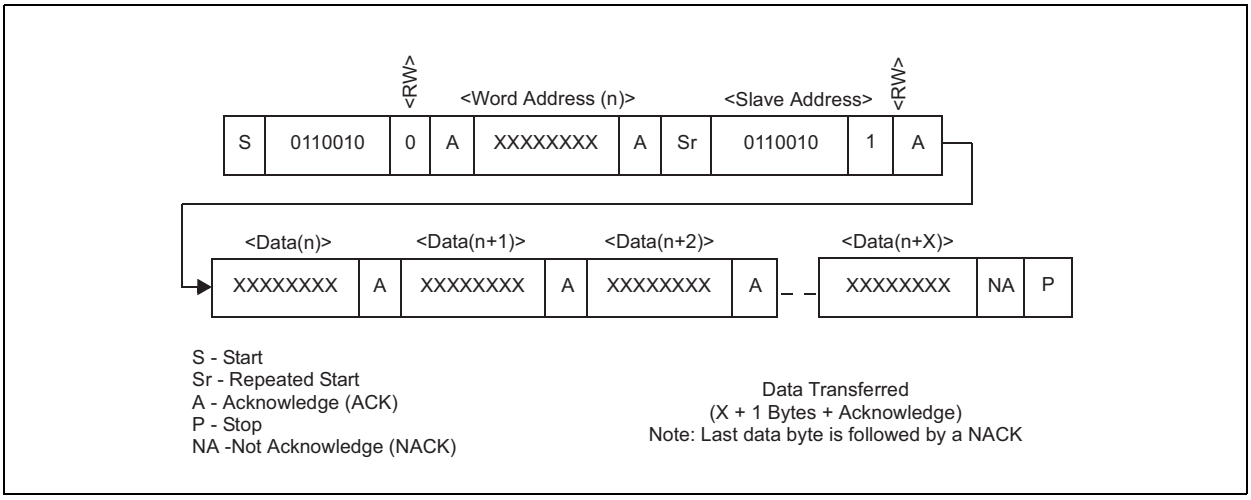


Figure 33:
Data Read (Write Pointer, Then Read) - Slave Receive and Transmit



Program Downloading

First the register page_select is set to the program page, which should be accessed. Then the program page (part of or full page) can be downloaded to the registers Cmd_0_MSB, Cmd_0_LSB, Cmd_1_MSB, Cmd_1_LSB to Cmd_F_MSB, Cmd_F_LSB (I^2C registers area 50h to 6Fh).

Figure 34:
Page_Select Register

Addr: 4Fh			Page_Select Register		
Bit	**Bit Name**	**Default**	**Access**	**Description**	
2:0	page_select	000b	R/W	Selects program page for download	
				000	page 0 -Addr 00h-0Fh
				001	page 1 -Addr 10h-1Fh
				010	page 2 -Addr 20h-2Fh
				011	page 3 -Addr 30h-3Fh
				100	page 4 -Addr 40h-4Fh
				101	page 5 -Addr 50h-5Fh
				110	Do not use
				111	Do not use

Register Set

The AS3661 is controlled by a set of registers through the two wire serial interface port. Some register bits are reserved for future use. Figure 35 below lists device registers, their addresses and their abbreviations. A more detailed description is given in Control Register Details.

Figure 35:
Description of Registers

Hex Address	Register Name	Bit(s)	Type	Default Value After Reset	Description
00	ENABLE / ENGINE CNTRL1	[6]	R/W	x0xxxxxx	CHIP_EN
					0 AS3661 not enabled
					1 AS3661 enabled
		[5:4]		xx00xxxx	ENGINE1_EXEC Engine 1 program execution control
		[3:2]		xxxx00xx	ENGINE2_EXEC Engine 2 program execution control
		[1:0]		xxxxxx00	ENGINE3_EXEC Engine 3 program execution control

Hex Address	Register Name	Bit(s)	Type	Default Value After Reset	Description
01	ENGINE CNTRL2	[5:4]	R/W	xx00xxxx	ENGINE1_MODE ENGINE 1 mode control
		[3:2]		xxxx00xx	ENGINE2_MODE ENGINE 2 mode control
		[1:0]		xxxxxx00	ENGINE3_MODE ENGINE 3 mode control
02	OUTPUT DIRECT/ RATIOMETRIC MSB	[0]	R/W	xxxxxxx0	LED9_RATIO_EN Enables ratiometric dimming for LED9 output
03	OUTPUT DIRECT/ RATIOMETRIC LSB	[7]	R/W	0xxxxxxx	LED8_RATIO_EN Enables ratiometric dimming for LED8 output
		[6]		x0xxxxxx	LED7_RATIO_EN Enables ratiometric dimming for LED7 output
		[5]		xx0xxxxx	LED6_RATIO_EN Enables ratiometric dimming for LED6 output
		[4]		xxx0xxxx	LED5_RATIO_EN Enables ratiometric dimming for LED5 output
		[3]		xxxx0xxx	LED4_RATIO_EN Enables ratiometric dimming for LED4 output
		[2]		xxxxx0xx	LED3_RATIO_EN Enables ratiometric dimming for LED3 output
		[1]		xxxxxx0x	LED2_RATIO_EN Enables ratiometric dimming for LED2 output
		[0]		xxxxxxx0	LED1_RATIO_EN Enables ratiometric dimming for LED1 output
04	OUTPUT ON/OFF CONTROL MSB	[0]	R/W	xxxxxxx1	LED9_ON ON/OFF Control for LED9 output

Hex Address	Register Name	Bit(s)	Type	Default Value After Reset	Description
05	OUTPUT ON / OFF CONTROL LSB	[7]	R/W	1xxxxxxx	LED8_ON ON/OFF Control for LED8 output
		[6]		x1xxxxxx	LED7_ON ON/OFF Control for LED7 output
		[5]		xx1xxxxx	LED6_ON ON/OFF Control for LED6 output
		[4]		xxx1xxxx	LED5_ON ON/OFF Control for LED5 output
		[3]		xxxx1xxx	LED4_ON ON/OFF Control for LED4 output
		[2]		xxxxx1xx	LED3_ON ON/OFF Control for LED3 output
		[1]		xxxxxx1x	LED2_ON ON/OFF Control for LED2 output
		[0]		xxxxxxx1	LED1_ON ON/OFF Control for LED1 output
06	LED1 CONTROL	[7:6]	R/W	00xxxxxx	MAPPING Mapping for LED1 output
		[5]		xx0xxxxx	LOG_EN Logarithmic dimming control for LED1
		[4:0]		xxx00000	TEMP COMP Temperature compensation control for LED1 output
07	LED2 CONTROL	[7:6]	R/W	00xxxxxx	MAPPING Mapping for LED2 output
		[5]		xx0xxxxx	LOG_EN Logarithmic dimming control for LED2 output
		[4:0]		xxx00000	TEMP COMP Temperature compensation control for LED2 output
08	LED3 CONTROL	[7:6]	R/W	00xxxxxx	MAPPING Mapping for LED3 output
		[5]		xx0xxxxx	LOG_EN Logarithmic dimming control for LED3 output
		[4:0]		xxx00000	TEMP COMP Temperature compensation control for LED3 output

Hex Address	Register Name	Bit(s)	Type	Default Value After Reset	Description
09	LED4 CONTROL	[7:6]	R/W	00xxxxxx	MAPPING Mapping for LED4 output
		[5]		xx0xxxxx	LOG_EN Logarithmic dimming control for LED4 output
		[4:0]		xxx00000	TEMP COMP Temperature compensation control for LED4 output
0A	LED5 CONTROL	[7:6]	R/W	00xxxxxx	MAPPING Mapping for LED5 output
		[5]		xx0xxxxx	LOG_EN Logarithmic dimming control for LED5 output
		[4:0]		xxx00000	TEMP COMP Temperature compensation control for LED5 output
0B	LED6 CONTROL	[7:6]	R/W	00xxxxxx	MAPPING Mapping for LED6 output
		[5]		xx0xxxxx	LOG_EN Logarithmic dimming control for LED6 output
		[4:0]		xxx00000	TEMP COMP Temperature compensation control for LED6 output
0C	LED7 CONTROL	[7:6]	R/W	00xxxxxx	MAPPING Mapping for LED7 output
		[5]		xx0xxxxx	LOG_EN Logarithmic dimming control for LED7 output
		[4:0]		xxx00000	TEMP COMP Temperature compensation control for LED7 output
0D	LED8 CONTROL	[7:6]	R/W	00xxxxxx	MAPPING Mapping for LED8 output
		[5]		xx0xxxxx	LOG_EN Logarithmic dimming control for LED8 output
		[4:0]		xxx00000	TEMP COMP Temperature compensation control for LED8 output

Hex Address	Register Name	Bit(s)	Type	Default Value After Reset	Description
0E	LED9 CONTROL	[7:6]	R/W	00xxxxxx	MAPPING Mapping for LED9 output
		[5]		xx0xxxxx	LOG_EN Logarithmic dimming control for LED9 output
		[4:0]		xxx00000	TEMP COMP Temperature compensation control for LED9 output
0F to 15		[7:0]			Reserved
16	LED1 PWM	[7:0]	R/W	00000000	PWM duty cycle control for LED1
17	LED2 PWM	[7:0]	R/W	00000000	PWM duty cycle control for LED2
18	LED3 PWM	[7:0]	R/W	00000000	PWM duty cycle control for LED3
19	LED4 PWM	[7:0]	R/W	00000000	PWM duty cycle control for LED4
1A	LED5 PWM	[7:0]	R/W	00000000	PWM duty cycle control for LED5
1B	LED6 PWM	[7:0]	R/W	00000000	PWM duty cycle control for LED6
1C	LED7 PWM	[7:0]	R/W	00000000	PWM duty cycle control for LED7
1D	LED8 PWM	[7:0]	R/W	00000000	PWM duty cycle control for LED8
1E	LED9 PWM	[7:0]	R/W	00000000	PWM duty cycle control for LED9
1F to 25		[7:0]			Reserved
26	LED1 CURRENT CONTROL	[7:0]	R/W	10101111	CURRENT LED1 output current control register. Default 17.5 mA (typ.)
27	LED2 CURRENT CONTROL	[7:0]	R/W	10101111	CURRENT LED2 output current control register. Default 17.5 mA (typ.)
28	LED3 CURRENT CONTROL	[7:0]	R/W	10101111	CURRENT LED3 output current control register. Default 17.5 mA (typ.)
29	LED4 CURRENT CONTROL	[7:0]	R/W	10101111	CURRENT LED4 output current control register. Default 17.5 mA (typ.)
2A	LED5 CURRENT CONTROL	[7:0]	R/W	10101111	CURRENT LED5 output current control register. Default 17.5 mA (typ.)
2B	LED6 CURRENT CONTROL	[7:0]	R/W	10101111	CURRENT LED6 output current control register. Default 17.5 mA (typ.)

Hex Address	Register Name	Bit(s)	Type	Default Value After Reset	Description
2C	LED7 CURRENT CONTROL	[7:0]	R/W	10101111	CURRENT LED7 output current control register. Default 17.5 mA (typ.)
0x2D	LED8 CURRENT CONTROL	[7:0]	R/W	10101111	CURRENT LED8 output current control register. Default 17.5 mA (typ.)
0x2E	LED9 CURRENT CONTROL	[7:0]	R/W	10101111	CURRENT LED9 output current control register. Default 17.5 mA (typ.)
2F to 35		[7:0]			Reserved
36	MISC	[7]	R/W	0xxxxxxx	VARIABLE_D_SEL Variable LED source selection
		[6]		x1xxxxxx	EN_AUTO_INCR Serial bus address auto increment enable
		[5]		xx0xxxxx	POWERSAVE_EN Powersave mode enable
		[4:3]		xxx00xxx	CP_MODE Charge pump gain selection
		[2]		xxxxx0xx	PWM_PS_EN PWM cycle powersave enable
		[1]		xxxxxx0x	CLK_DET_EN External clock detection
		[0]		xxxxxxx0	INT_CLK_EN Clock source selection
37	ENGINE1 PC	[6:0]	R/W	x0000000	PC Program counter for engine 1
38	ENGINE2 PC	[6:0]	R/W	x0000000	PC Program counter for engine 2
39	ENGINE3 PC	[6:0]	R/W	x0000000	PC Program counter for engine 3

Hex Address	Register Name	Bit(s)	Type	Default Value After Reset	Description
3A	STATUS / INTERRUPT	[7]	R	0xxxxxxx	LEDTEST_MEAS_DONE Indicates when the LED test measurement is done.
		[6]		x1xxxxxx	MASK_BUSY Mask bit for interrupts generated by STARTUP_BUSY or ENGINE_BUSY
		[5]		xx0xxxxx	STARTUP_BUSY This bit indicates that the start-up sequence is running
		[4]		xxx0xxxx	ENGINE_BUSY This bit indicates that a program execution engine is clearing internal registers
		[3]		xxxx0xxx	EXT_CLK_USED Indicates when external clock signal is in use
		[2]		xxxxx0xx	ENG1_INT Interrupt bit for program execution engine 1
		[1]		xxxxxx0x	ENG2_INT Interrupt bit for program execution engine 2
		[0]		xxxxxxx0	ENG3_INT Interrupt bit for program execution engine 3
3B	GPO	[2]	R/W	xxxxx0xx	INT_CONF INT pin can be configured to function as a GPO with this bit
		[1]		xxxxxx0x	GPO pin control
		[0]		xxxxxxx0	INT_GPO GPO pin control for INT pin (when INT_CONF is set "1")
3C	VARIABLE	[7:0]	R/W	00000000	VARIABLE Global 8-bit variable
3D	RESET	[7:0]	R/W	00000011	RESET Writing 11111111 into this register resets the AS3661

Hex Address	Register Name	Bit(s)	Type	Default Value After Reset	Description
3E	TEMP ADC CONTROL	[7]	R	0xxxxxxx	TEMP_MEAS_BUSY Indicates when temperature measurement is active
		[2]	R/W	xxxxx0xx	EN_TEMP_SENSOR Reads the internal temperature sensor once
		[1]		xxxxxx0x	CONTINUOUS_CONV Continuous temperature measurement selection
		[0]		xxxxxxx0	SEL_EXT_TEMP Internal/external temperature sensor selection
3F	TEMPERATURE READ	[7:0]	R	00011001	TEMPERATURE Bits for temperature information
40	TEMPERATURE WRITE	[7:0]	R/W	0000000	TEMPERATURE Bits for temperature information
41	LED TEST CONTROL	[7]	R/W	0xxxxxxx	EN_LED_TEST_ADC
		[6]		x0xxxxxx	EN_LED_TEST_INT
		[5]		xx0xxxxx	CONTINUOUS_CONV Continuous LED test measurement selection
		[4:0]		xxx00000	LED_TEST_CTRL Control bits for LED test
42	LED TEST ADC	[7:0]	R	N/A	LED_TEST_ADC LED test result
43		[7:0]			Reserved
44		[7:0]			Reserved
45	ENGINE1 VARIABLE A	[7:0]	R	00000000	VARIABLE FOR ENGINE1
46	ENGINE2 VARIABLE A	[7:0]	R	00000000	VARIABLE FOR ENGINE2
47	ENGINE3 VARIABLE A	[7:0]	R	00000000	VARIABLE FOR ENGINE3
48	MASTER FADER1	[7:0]	R/W	00000000	MASTER FADER 1
49	MASTER FADER2	[7:0]	R/W	00000000	MASTER FADER 2
4A	MASTER FADER3	[7:0]	R/W	00000000	MASTER FADER 3
4B		[7:0]			Reserved

Hex Address	Register Name	Bit(s)	Type	Default Value After Reset	Description
4C	ENG1 PROG START ADDR	[6:0]	R/W	x0000000	Engine 1 program start address
4D	ENG2 PROG START ADDR	[6:0]	R/W	x0001000	Engine 2 program start address
4E	ENG3 PROG START ADDR	[6:0]	R/W	x0010000	Engine 3 program start address
4F	PROG MEM PAGE SEL	[2:0]	R/W	xxxxx000	PAGE_SEL
50	PROGRAM MEMORY 00H/10H/20H/30H / 40H/50H	[15:8]	R/W	00000000	CMD Every Instruction is 16–bit width. The AS3661 can store 96 instructions. Each instruction consists of 16 bits. Because one register has only 8 bits, one instruction requires two register addresses. In order to reduce program load time the AS3661 supports address auto-incrementation. Register address is incremented after each 8 data bits. Thus the whole program memory page can be written in one serial bus write sequence.
51		[7:0]		00000000	
52	PROGRAM MEMORY 01H/11H/21H/31H / 41H/51H	[15:8]	R/W	00000000	
53		[7:0]		00000000	
54	PROGRAM MEMORY 02H/12H/22H/32H / 42H/52H	[15:8]	R/W	00000000	
55		[7:0]		00000000	
56	PROGRAM MEMORY 03H/13H/23H/33H / 43H/53H	[15:8]	R/W	00000000	
57		[7:0]		00000000	

Hex Address	Register Name	Bit(s)	Type	Default Value After Reset	Description
58	PROGRAM MEMORY 04H/14H/24H/34H / 44H/54H	[15:8]	R/W	00000000	CMD Every Instruction is 16–bit width. The AS3661 can store 96 instructions. Each instruction consists of 16 bits. Because one register has only 8 bits, one instruction requires two register addresses. In order to reduce program load time the AS3661 supports address auto-incrementation. Register address is incremented after each 8 data bits. Thus the whole program memory page can be written in one serial bus write sequence.
59		[7:0]		00000000	
5A	PROGRAM MEMORY 05H/15H/25H/35H / 45H/55H	[15:8]	R/W	00000000	
5B		[7:0]		00000000	
5C	PROGRAM MEMORY 06H/16H/26H/36H / 46H/56H	[15:8]	R/W	00000000	
5D		[7:0]		00000000	
5E	PROGRAM MEMORY 07H/17H/27H/37H / 47H/57H	[15:8]	R/W	00000000	
5F		[7:0]		00000000	
60	PROGRAM MEMORY 08H/18H/28H/38H / 48H/58H	[15:8]	R/W	00000000	
61		[7:0]		00000000	
62	PROGRAM MEMORY 09H/19H/29H/39H / 49H/59H	[15:8]	R/W	00000000	
63		[7:0]		00000000	
64	PROGRAM MEMORY 0AH/1AH/2AH/3AH/ 4AH/5AH	[15:8]	R/W	00000000	
65		[7:0]		00000000	
66	PROGRAM MEMORY 0BH/1BH/2BH/3BH/ 4BH/5BH	[15:8]	R/W	00000000	
67		[7:0]		00000000	
68	PROGRAM MEMORY 0CH/1CH/2CH/ 3CH/4CH/5CH	[15:8]	R/W	00000000	
69		[7:0]		00000000	
6A	PROGRAM MEMORY 0DH/1DH/2DH/36D/ 46D/5DH	[15:8]	R/W	00000000	
6B		[7:0]		00000000	

Hex Address	Register Name	Bit(s)	Type	Default Value After Reset	Description
6C	PROGRAM MEMORY 0EH/1EH/2EH/3EH / 4EH/5EH	[15:8]	R/W	00000000	CMD Every Instruction is 16–bit width. The AS3661 can store 96 instructions. Each instruction consists of 16 bits. Because one register has only 8 bits, one instruction requires two register addresses. In order to reduce program load time the AS3661 supports address auto-incrementation. Register address is incremented after each 8 data bits. Thus the whole program memory page can be written in one serial bus write sequence.
6D		[7:0]		00000000	
6E	PROGRAM MEMORY 0FH/1FH/2FH/3FH / 4FH/5FH	[15:8]	R/W	00000000	
6F		[7:0]		00000000	
70	ENG1 MAPPING MSB	[7]	R	0xxxxxxx	GPO Engine 1 mapping information, GPO pin
		[0]		xxxxxxx0	D9 Engine 1 mapping information, D9 output
71	ENG1 MAPPING LSB	[7]	R	0xxxxxxx	LED8 Engine 1 mapping information, LED8 output
		[6]		x0xxxxxx	LED7 Engine 1 mapping information, LED7 output
		[5]		xx0xxxxx	LED6 Engine 1 mapping information, LED6 output
		[4]		xxx0xxxx	LED5 Engine 1 mapping information, LED5 output
		[3]		xxxx0xxx	LED4 Engine 1 mapping information, LED4 output
		[2]		xxxxx0xx	LED3 Engine 1 mapping information, LED3 output
		[1]		xxxxxx0x	LED2 Engine 1 mapping information, LED2 output
		[0]		xxxxxxx0	LED1 Engine 1 mapping information, LED1 output

Hex Address	Register Name	Bit(s)	Type	Default Value After Reset	Description
72	ENG2 MAPPING MSB	[7]	R	0xxxxxxx	GPO Engine 2 mapping information, GPO pin
		[0]		xxxxxxx0	LED9 Engine 2 mapping information, D9 output
73	ENG2 MAPPING LSB	[7]	R	0xxxxxxx	LED8 Engine 2 mapping information, LED8 output
		[6]		x0xxxxxx	LED7 Engine 2 mapping information, LED7 output
		[5]		xx0xxxxx	LED6 Engine 2 mapping information, LED6 output
		[4]		xxx0xxxx	LED5 Engine 2 mapping information, LED5 output
		[3]		xxxx0xxx	LED4 Engine 2 mapping information, LED4 output
		[2]		xxxxx0xx	LED3 Engine 2 mapping information, LED3 output
		[1]		xxxxxx0x	LED2 Engine 2 mapping information, LED2 output
		[0]		xxxxxxx0	LED1 Engine 2 mapping information, LED1 output
74	ENG3 MAPPING MSB	[7]	R	0xxxxxxx	GPO Engine 3 mapping information, GPO pin
		[0]		xxxxxxx0	LED9 Engine 3 mapping information, LED9 output

Hex Address	Register Name	Bit(s)	Type	Default Value After Reset	Description
75	ENG3 MAPPING LSB	[7]	R	0xxxxxxx	LED8 Engine 3 mapping information, LED8 output
		[6]		x0xxxxxx	LED7 Engine 3 mapping information, LED7 output
		[5]		xx0xxxxx	LED6 Engine 3 mapping information, LED6 output
		[4]		xxx0xxxx	LED5 Engine 3 mapping information, LED5 output
		[3]		xxxx0xxx	LED4 Engine 3 mapping information, LED4 output
		[2]		xxxxx0xx	LED3 Engine 3 mapping information,LED3 output
		[1]		xxxxxx0x	LED2 Engine 3 mapping information, LED2 output
		[0]		xxxxxxx0	LED1 Engine 3 mapping information, LED1 output

Hex Address	Register Name	Bit(s)	Type	Default Value After Reset	Description
76	GAIN CHANGE CTRL	[7:6]	R/W	00xxxxxx	TRESHOLD Threshold voltage (typ.) 00 400mV 01 300mV 10 200mV 11 100mV
		[5]	R/W	xx0xxxxx	ADAPTIVE_TRESH_EN Activates adaptive threshold.
		[4:3]	R/W	xxx00xxx	TIMER 00 5ms 01 10ms 10 50ms 11 Infinite
		[2]	R/W	xxxxx0xx	FORCE_1x Activates 1.5x to 1x timer

Control Register Details

ENABLE/ ENGINE CONTROL1

This register controls the startup of the chip and the program execution modes for each program execution engine.

Figure 36:
ENABLE / ENGINE CNTR 1 Register

Register: 0x00		ENABLE / ENGINE CNTR1		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
6	CHIP_EN	0	R/W	**0: Standby mode is entered. Still, control registers can be written or read, excluding bits [5:0] in reg 00 (this register), registers 16h to 1E (LED PWM registers) and 37h to 39h (program counters).** 1: internal startup sequence powers up all the needed internal blocks and the device enters normal mode.

Register: 0x00		ENABLE / ENGINE CNTR1		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
5:4	ENGINE1_EXEC	00	R/W	The engine 1 program execution control register bits define how the program is executed. Program start address can be programmed to program counter (PC) register 0 × 37. **00: Hold causes the execution engine to finish the current instruction and then stop. Program counter (PC) can be read or written only in this mode.** 01: Execute the instruction at the location pointed by the PC, increment the PC by one and then reset ENG1_EXEC bits to 00 (i.e. enter hold). 10: Start program execution from the location pointed by the PC. This mode is also called “Free Run” mode. 11: Execute the instruction pointed by the current PC value and reset ENG1_EXEC to 00 (i.e. enter hold). The difference between step and execute once is that execute once does not increment the PC.
3:2	ENGINE2_EXEC	00	R/W	The engine 2 program execution control register bits define how the program is executed. Program start address can be programmed to program counter (PC) register 0 × 38. **00: Hold causes the execution engine to finish the current instruction and then stop. Program counter (PC) can be read or written only in this mode.** 01: Execute the instruction at the location pointed by the PC, increment the PC by one and then reset ENG2_EXEC bits to 00 (i.e. enter hold). 10: Start program execution from the location pointed by the PC. This mode is also called “Free Run” mode. 11: Execute the instruction pointed by the current PC value and reset ENG2_EXEC to 00 (i.e. enter hold). The difference between step and execute once is that execute once does not increment the PC.

Register: 0x00		ENABLE / ENGINE CNTR1		
Bit	Bit Name	Default	Access	Bit Description
1:0	ENGINE3_EXEC	00	R/W	The engine 3 program execution control register bits define how the program is executed. Program start address can be programmed to program counter (PC) register 0 × 39. **00: Hold causes the execution engine to finish the current instruction and then stop. Program counter (PC) can be read or written only in this mode.** 01: Executes the instruction at the location pointed by the PC, increment the PC by one and then reset ENG3_EXEC bits to 00 (i.e. enter hold). 10: Start program execution from the location pointed by the PC. This mode is also called “Free Run” mode. 11: Execute the instruction pointed by the current PC value and reset ENG3_EXEC to 00 (i.e. enter hold). The difference between step and execute once is that execute once does not increment the PC.

ENGINE CNTRL2

The AS3661 supports up to four different operation modes which are defined in these registers.

Disabled: Engines can be configured to disabled mode each one separately.

Load program: Writing to program memory is allowed only when the engine is in load program operation mode and engine busy bit (reg 3A) is not set. Serial bus master should check the busy bit before writing to program memory. All the three engines are in hold while one or more engines are in load program mode. PWM values are frozen, also. Program execution continues when all the engines are out of load program mode. Load program mode resets the program counter of the respective engine. Load program mode can be entered from the disabled mode only. Entering load program mode from the run program mode is not allowed.

Run Program: Run program mode executes the instructions stored in the program memory. Execution register (ENG1_EXEC etc.) bits define how the program is executed (hold, step, free run or execute once). Program start address can be programmed to the Program Counter (PC) register. The Program Counter is reset to zero when the PC’s upper limit value is reached.

Halt: Instruction execution aborts immediately and engine operation halts.

Figure 37:
ENGINE CNTRL2 Register

Register: 0x01		ENGINE CNTRL2		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
5:4	ENGINE1_MODE	00	R/W	**00: Disabled** 01: Load program to SRAM, reset engine 1 PC 10: Run program as defined by ENGINE1_EXEC bits 11: Halts the engine
3:2	ENGINE2_MODE	00	R/W	**00: Disabled** 01: Load program to SRAM, reset engine 2 PC 10: Run program as defined by ENGINE2_EXEC bits 11: Halts the engine
1:0	ENGINE3_MODE	00	R/W	**00: Disabled** 01: Load program to SRAM, reset engine 3 PC 10: Run program as defined by ENGINE3_EXEC bits 11: Halts the engine

OUTPUT DIRECT/RATIOMETRIC MSB and LSB

A particular feature of the AS3661 is the ratiometric up/down dimming of the RGB-LEDs. In other words, the LED driver PWM output will vary in a ratiometric manner. By a ratiometric approach the emitted color of an RGB–LED remains the same regardless of the initial magnitudes of the R/G/B PWM outputs. For example, if the PWM output of the red LED output is doubled, the output of green LED is doubled also.

Figure 38:
OUTPUT DIRECT / RATIOMETRIC MSB Register

Register: 0x02		OUTPUT DIRECT/RATIOMETRIC MSB		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
0	LED9_RATIO_EN	0	R/W	**0: Disables ratiometric dimming for LED9 output.** 1: enables ratiometric dimming for LED9 output.

Figure 39:
OUTPUT DIRECT / RATIOMETRIC LSB Register

Register: 0x03		OUTPUT DIRECT/RATIOMETRIC LSB		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7	LED8_RATIO_EN	0	R/W	**0: Disables ratiometric dimming for LED8 output.** 1: enables ratiometric dimming for LED9 output.

Register: 0x03		OUTPUT DIRECT/RATIOMETRIC LSB		
Bit	Bit Name	Default	Access	Bit Description
6	LED7_RATIO_EN	0	R/W	**0: Disables ratiometric dimming for LED7 output.** 1: enables ratiometric dimming for LED9 output.
5	LED6_RATIO_EN	0	R/W	**0: Disables ratiometric dimming for LED6 output.** 1: enables ratiometric dimming for LED9 output.
4	LED5_RATIO_EN	0	R/W	**0: Disables ratiometric dimming for LED5 output.** 1: enables ratiometric dimming for LED9 output.
3	LED4_RATIO_EN	0	R/W	**0: Disables ratiometric dimming for LED4 output.** 1: enables ratiometric dimming for LED9 output.
2	LED3_RATIO_EN	0	R/W	**0: Disables ratiometric dimming for LED3 output.** 1: enables ratiometric dimming for LED9 output.
1	LED2_RATIO_EN	0	R/W	**0: Disables ratiometric dimming for LED2 output.** 1: enables ratiometric dimming for LED9 output.
0	LED1_RATIO_EN	0	R/W	**0: Disables ratiometric dimming for LED1 output.** 1: enables ratiometric dimming for LED9 output.

OUTPUT ON/OFF CONTROL MSB and LSB

The following two registers allow the user to switch all nine current sources independently from each other ON and OFF. Please mind that this selection will be overridden if a current source is selected by one of the program execution engines.

Figure 40:
OUTPUT ON/OFF CONTROL MSB Register

Register: 0x04		OUTPUT ON/OFF CONTROL MSB		
Bit	Bit Name	Default	Access	Bit Description
0	LED9_ON	0	R/W	**0: LED9 output OFF.** 1: LED9 output ON.

Figure 41:
OUTPUT ON/OFF CONTROL LSB Register

Register: 0x05		OUTPUT ON/OFF CONTROL LSB		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7	LED8_ON	0	R/W	**0: LED8 output OFF.** 1: LED8 output ON.
6	LED7_ON	0	R/W	**0: LED7 output OFF.** 1: LED7 output ON.
5	LED6_ON	0	R/W	**0: LED6 output OFF.** 1: LED6 output ON.
4	LED5_ON	0	R/W	**0: LED5 output OFF.** 1: LED5 output ON.
3	LED4_ON	0	R/W	**0: LED4 output OFF.** 1: LED4 output ON.
2	LED3_ON	0	R/W	**0: LED3 output OFF.** 1: LED3 output ON.
1	LED2_ON	0	R/W	**0: LED2 output OFF.** 1: LED2 output ON.
0	LED1_ON	0	R/W	**0: LED1 output OFF.** 1: LED1 output ON.

LEDx Control

These registers are used to assign the any current source output to the MASTER FADER group 1, 2, or 3, or none of them. Also, the registers set the slope of the current sources output temperature compensation line and selects between linear and logarithmic PWM brightness adjustment. By using logarithmic PWM-scale the visual effect looks like linear. When the logarithmic adjustment is enabled, the chip handles internally PWM values with 12-bit resolution. This allows very fine-grained PWM control at low PWM duty cycles. If a MASTER FADER is selected for an output, the duty cycle on the output will be LED1 PWM register value (address 0x16) multiplied with the value in the MASTER FADER register.

Besides the LED mapping and linear or logarithmic selection it is also possible to do a temperature compensation for each output separately. The PWM duty cycle at temperature T (in centigrade) can be obtained as follows:

$PWM_F = [PWM_S - (25 - T) * slope * PWMs] / 2$,

where PWMF is the final duty cycle at temperature T, PWMs is the set PWM duty cycle (PWM duty cycle is set in registers 16H to 1EH) and the value of the correction factor is obtained from Figure 36.

For example, if the set PWM duty cycle in register 16H is 90%, temperature T is -10°C and the chosen slope is +1.5 1/ °C, the final duty cycle PWMF for LED1 output will be

[90% - (25°C - (-10°C))* 1.5 1/°C * 90%]/2 =
[90 % - 35 * 1.5 * 90%]/2 = 21.4%.

Default setting 00000 means that the temperature compensation is non-active and the PWM output (0 to 100%) is set solely by PWM registers LED1 PWM to LED9 PWM.

Figure 42:
LED1 CONTROL Register

Register: 0x06		LED1 CONTROL		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:6	LED1_MAPPING	00	R/W	This register defines the mapping of LED1 output to the master faders. The faders can either be used for dimming several LEDs in parallel or for ratiometric control of the output. **00: no master fader selected** 01: MASTER FADER 1 controls LED1 output 10: MASTER FADER 2 controls LED1 output 11: MASTER FADER 3 controls LED1 output
5	LED1_LOG_EN		R/W	This bit is effective for both, program execution engine and direct PWM control. **0: linear adjustment.** 1: logarithmic adjustment.

Register: 0x06		LED1 CONTROL		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
4:0	LED1_TEMP_COMP	0 0000	R/W	The reference temperature is 25°C (i.e. the temper ature at which all slope settings have no effect) and the temperature coefficient (slope) can be set in 0.11/°C steps to any value between -1.5 1/°C and +1.5 1/°C, with a default to 0.0 1/°C 11111: -1.5 1/°C 11110: -1.4 1/°C ... **00000: temperature compensation not activated** ... 01110: +1.4 1/°C 01111: +1.5 1/°C

Figure 43:
LED2 CONTROL Register

Register: 0x07		LED2 CONTROL		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:6	LED2_MAPPING	00	R/W	This register defines the mapping of LED2 output to the master faders. The faders can either be used for dimming several LEDs in parallel or for ratiometric control of the output. **00: no master fader selected** 01: MASTER FADER 1 controls LED2 output 10: MASTER FADER 2 controls LED2 output 11: MASTER FADER 3 controls LED2 output
5	LED2_LOG_EN	0	R/W	This bit is effective for both, program execution engine and direct PWM control. **0: linear adjustment.** 1: logarithmic adjustment.

Register: 0x07		LED2 CONTROL		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
4:0	LED2_TEMP_COMP	0 0000	R/W	The reference temperature is 25°C (i.e. the temper ature at which all slope settings have no effect) and the temperature coefficient (slope) can be set in 0.11/°C steps to any value between -1.5 1/°C and +1.5 1/°C, with a default to 0.0 1/°C 11111: -1.5 1/°C 11110: -1.4 1/°C ... **00000: temperature compensation not activated** ... 01110: +1.4 1/°C 01111: +1.5 1/°C

Figure 44:
LED3 CONTROL Register

Register: 0x08		LED3 CONTROL		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:6	LED3_MAPPING	00	R/W	This register defines the mapping of LED3 output to the master faders. The faders can either be used for dimming several LEDs in parallel or for ratiometric control of the output. **00: no master fader selected** 01: MASTER FADER 1 controls LED3 output 10: MASTER FADER 2 controls LED3 output 11: MASTER FADER 3 controls LED3 output
5	LED3_LOG_EN	0	R/W	This bit is effective for both, program execution engine and direct PWM control. **0: linear adjustment.** 1: logarithmic adjustment.

Register: 0x08		LED3 CONTROL		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
4:0	LED3_TEMP_COMP	0 0000	R/W	The reference temperature is 25°C (i.e. the temper ature at which all slope settings have no effect) and the temperature coefficient (slope) can be set in 0.11/°C steps to any value between -1.5 1/°C and +1.5 1/°C, with a default to 0.0 1/°C 11111: -1.5 1/°C 11110: -1.4 1/°C ... **00000: temperature compensation not activated** ... 01110: +1.4 1/°C 01111: +1.5 1/°C

Figure 45:
LED4 CONTROL Register

Register: 0x09		LED4 CONTROL		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:6	LED4_MAPPING	00	R/W	This register defines the mapping of LED4 output to the master faders. The faders can either be used for dimming several LEDs in parallel or for ratiometric control of the output. **00: no master fader selected** 01: MASTER FADER 1 controls LED4 output 10: MASTER FADER 2 controls LED4 output 11: MASTER FADER 3 controls LED4 output
5	LED4_LOG_EN	0	R/W	This bit is effective for both, program execution engine and direct PWM control. **0: linear adjustment.** 1: logarithmic adjustment.

Register: 0x09		LED4 CONTROL		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
4:0	LED4_TEMP_COMP	0 0000	R/W	The reference temperature is 25°C (i.e. the temper ature at which all slope settings have no effect) and the temperature coefficient (slope) can be set in 0.11/°C steps to any value between -1.5 1/°C and +1.5 1/°C, with a default to 0.0 1/°C 11111: -1.5 1/°C 11110: -1.4 1/°C ... **00000: temperature compensation not activated** ... 01110: +1.4 1/°C 01111: +1.5 1/°C

Figure 46:
LED5 CONTROL Register

Register: 0x0A		LED5 CONTROL		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:6	LED5_MAPPING	00	R/W	This register defines the mapping of LED5 output to the master faders. The faders can either be used for dimming several LEDs in parallel or for ratiometric control of the output. **00: no master fader selected** 01: MASTER FADER 1 controls LED5 output 10: MASTER FADER 2 controls LED5 output 11: MASTER FADER 3 controls LED5 output
5	LED5_LOG_EN	0	R/W	This bit is effective for both, program execution engine and direct PWM control. **0: linear adjustment.** 1: logarithmic adjustment.

Register: 0x0A		LED5 CONTROL		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
4:0	LED5_TEMP_COMP	0 0000	R/W	The reference temperature is 25°C (i.e. the temper ature at which all slope settings have no effect) and the temperature coefficient (slope) can be set in 0.11/°C steps to any value between -1.5 1/°C and +1.5 1/°C, with a default to 0.0 1/°C 11111: -1.5 1/°C 11110: -1.4 1/°C ... **00000: temperature compensation not activated** ... 01110: +1.4 1/°C 01111: +1.5 1/°C

Figure 47:
LED6 CONTROL Register

Register: 0x0B		LED6 CONTROL		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:6	LED6_MAPPING	00	R/W	This register defines the mapping of LED6 output to the master faders. The faders can either be used for dimming several LEDs in parallel or for ratiometric control of the output. **00: no master fader selected** 01: MASTER FADER 1 controls LED6 output 10: MASTER FADER 2 controls LED6 output 11: MASTER FADER 3 controls LED6 output
5	LED6_LOG_EN	0	R/W	This bit is effective for both, program execution engine and direct PWM control. **0: linear adjustment.** 1: logarithmic adjustment.

Register: 0x0B		LED6 CONTROL		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
4:0	LED6_TEMP_COMP	0 0000	R/W	The reference temperature is 25°C (i.e. the temper ature at which all slope settings have no effect) and the temperature coefficient (slope) can be set in 0.11/°C steps to any value between -1.5 1/°C and +1.5 1/°C, with a default to 0.0 1/°C 11111: -1.5 1/°C 11110: -1.4 1/°C ... **00000: temperature compensation not activated** ... 01110: +1.4 1/°C 01111: +1.5 1/°C

Figure 48:
LED7 CONTROL Register

Register: 0x0C		LED7 CONTROL		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:6	LED7_MAPPING	00	R/W	This register defines the mapping of LED7 output to the master faders. The faders can either be used for dimming several LEDs in parallel or for ratiometric control of the output. **00: no master fader selected** 01: MASTER FADER 1 controls LED7 output 10: MASTER FADER 2 controls LED7 output 11: MASTER FADER 3 controls LED7 output
5	LED7_LOG_EN	0	R/W	This bit is effective for both, program execution engine and direct PWM control. **0: linear adjustment.** 1: logarithmic adjustment.

Register: 0x0C		LED7 CONTROL		
Bit	Bit Name	Default	Access	Bit Description
4:0	LED7_TEMP_COMP	0 0000	R/W	The reference temperature is 25°C (i.e. the temper ature at which all slope settings have no effect) and the temperature coefficient (slope) can be set in 0.11/°C steps to any value between -1.5 1/°C and +1.5 1/°C, with a default to 0.0 1/°C 11111: -1.5 1/°C 11110: -1.4 1/°C ... **00000: temperature compensation not activated** ... 01110: +1.4 1/°C 01111: +1.5 1/°C

Figure 49:
LED8 CONTROL Register

Register: 0x0D		LED8 CONTROL		
Bit	Bit Name	Default	Access	Bit Description
7:6	LED8_MAPPING	00	R/W	This register defines the mapping of LED8 output to the master faders. The faders can either be used for dimming several LEDs in parallel or for ratiometric control of the output. **00: no master fader selected** 01: MASTER FADER 1 controls LED8 output 10: MASTER FADER 2 controls LED8 output 11: MASTER FADER 3 controls LED8 output
5	LED8_LOG_EN	0	R/W	This bit is effective for both, program execution engine and direct PWM control. **0: linear adjustment.** 1: logarithmic adjustment.

Register: 0x0D		LED8 CONTROL		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
4:0	LED8_TEMP_COMP	0 0000	R/W	The reference temperature is 25°C (i.e. the temper ature at which all slope settings have no effect) and the temperature coefficient (slope) can be set in 0.11/°C steps to any value between -1.5 1/°C and +1.5 1/°C, with a default to 0.0 1/°C 11111: -1.5 1/°C 11110: -1.4 1/°C ... **00000: temperature compensation not activated** ... 01110: +1.4 1/°C 01111: +1.5 1/°C

Figure 50:
LED9 CONTROL Register

Register: 0x0E		LED9 CONTROL		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:6	LED9_MAPPING	00	R/W	This register defines the mapping of LED9 output to the master faders. The faders can either be used for dimming several LEDs in parallel or for ratiometric control of the output. **00: no master fader selected** 01: MASTER FADER 1 controls LED9 output 10: MASTER FADER 2 controls LED9 output 11: MASTER FADER 3 controls LED9 output
5	LED9_LOG_EN	0	R/W	This bit is effective for both, program execution engine and direct PWM control. **0: linear adjustment.** 1: logarithmic adjustment.

Register: 0x0E		LED9 CONTROL		
Bit	Bit Name	Default	Access	Bit Description
4:0	LED9_TEMP_COMP	0 0000	R/W	The reference temperature is 25°C (i.e. the temper ature at which all slope settings have no effect) and the temperature coefficient (slope) can be set in 0.11/°C steps to any value between -1.5 1/°C and +1.5 1/°C, with a default to 0.0 1/°C 11111: -1.5 1/°C 11110: -1.4 1/°C ... **00000: temperature compensation not activated** ... 01110: +1.4 1/°C 01111: +1.5 1/°C

LED x PWM

This is the PWM duty cycle control for LED1 to LED9 output. The PWM registers are effective during direct control operation. Direct PWM control is active after power up by default.

Note(s):

- Serial bus address auto increment is not supported for register addresses from 16 to 1E.
- If the temperature compensation is active, the maximum PWM duty cycle is 50% at 25°C. This is required to allow enough headroom for temperature compensation over the temperature range -40°C to 90°C.

Figure 51:
LED1 PWM Register

Register: 0x16		LED1 PWM		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:0	LED1_PWM	0000 0000	R/W	This register controls the duty cycle of LED1 PWM output. **0000 0000: 0% Duty Cycle** 0000 0001: 0.3921% Duty Cycle 0000 0010: 0.7843% Duty Cycle 0000 0011: 1.1765% Duty Cycle .... 1111 1111: 100% Duty Cycle

Figure 52:
LED2 PWM Register

Register: 0x17		LED2 PWM		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:0	LED2_PWM	0000 0000	R/W	This register controls the duty cycle of LED2 PWM output. **0000 0000: 0% Duty Cycle** 0000 0001: 0.3921% Duty Cycle 0000 0010: 0.7843% Duty Cycle 0000 0011: 1.1765% Duty Cycle .... 1111 1111: 100% Duty Cycle

Figure 53:
LED3 PWM Register

Register: 0x18		LED3 PWM		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:0	LED3_PWM	0000 0000	R/W	This register controls the duty cycle of LED3 PWM output. **0000 0000: 0% Duty Cycle** 0000 0001: 0.3921% Duty Cycle 0000 0010: 0.7843% Duty Cycle 0000 0011: 1.1765% Duty Cycle .... 1111 1111: 100% Duty Cycle

Figure 54:
LED4 PWM Register

Register: 0x19		LED4 PWM		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:0	LED4_PWM	0000 0000	R/W	This register controls the duty cycle of LED4 PWM output. **0000 0000: 0% Duty Cycle** 0000 0001: 0.3921% Duty Cycle 0000 0010: 0.7843% Duty Cycle 0000 0011: 1.1765% Duty Cycle .... 1111 1111: 100% Duty Cycle

Figure 55:
LED5 PWM Register

Register: 0x1A		LED5 PWM		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:0	LED5_PWM	0000 0000	R/W	This register controls the duty cycle of LED5 PWM output. **0000 0000: 0% Duty Cycle** 0000 0001: 0.3921% Duty Cycle 0000 0010: 0.7843% Duty Cycle 0000 0011: 1.1765% Duty Cycle .... 1111 1111: 100% Duty Cycle

Figure 56:
LED6 PWM Register

Register: 0x1B		LED6 PWM		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:0	LED6_PWM	0000 0000	R/W	This register controls the duty cycle of LED6 PWM output. **0000 0000: 0% Duty Cycle** 0000 0001: 0.3921% Duty Cycle 0000 0010: 0.7843% Duty Cycle 0000 0011: 1.1765% Duty Cycle .... 1111 1111: 100% Duty Cycle

Figure 57:
LED7 PWM Register

Register: 0x1C		LED7 PWM		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:0	LED7_PWM	0000 0000	R/W	This register controls the duty cycle of LED7 PWM output. **0000 0000: 0% Duty Cycle** 0000 0001: 0.3921% Duty Cycle 0000 0010: 0.7843% Duty Cycle 0000 0011: 1.1765% Duty Cycle .... 1111 1111: 100% Duty Cycle

Figure 58:
LED8 PWM Register

Register: 0x1D		LED8 PWM		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:0	LED8_PWM	0000 0000	R/W	This register controls the duty cycle of LED8 PWM output. **0000 0000: 0% Duty Cycle** 0000 0001: 0.3921% Duty Cycle 0000 0010: 0.7843% Duty Cycle 0000 0011: 1.1765% Duty Cycle .... 1111 1111: 100% Duty Cycle

Figure 59:
LED9 PWM Register

Register: 0x1E		LED9 PWM		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:0	LED9_PWM	0000 0000	R/W	This register controls the duty cycle of LED9 PWM output. **0000 0000: 0% Duty Cycle** 0000 0001: 0.3921% Duty Cycle 0000 0010: 0.7843% Duty Cycle 0000 0011: 1.1765% Duty Cycle .... 1111 1111: 100% Duty Cycle

LEDx CURRENT CONTROL

With the following register it is possible to control the output current of each current source separately. The resolution of the current sources is 8-bit which gives a step size is 100 µA with a maximum output current of 25.5mA per current source.

Figure 60:
LED1 CURRENT CONTROL Register

Register: 0x26		LED1 CURRENT CONTROL		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:0	LED1_CURRENT	1010 1111	R/W	This register controls the output current of current source LED1 in 100µA steps from 0µA up to 25.5mA. 0000 0000: 0mA 0000 0001: 0.1mA ... **1010 1111: 17.5mA** ... 1111 1110: 25.4mA 1111 1111: 25.5mA

Figure 61:
LED2 CURRENT CONTROL Register

Register: 0x27		LED2 CURRENT CONTROL		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:0	LED2_CURRENT	1010 1111	R/W	This register controls the output current of current source LED2 in 100µA steps from 0µA up to 25.5mA. 0000 0000: 0mA 0000 0001: 0.1mA ... **1010 1111: 17.5mA** ... 1111 1110: 25.4mA 1111 1111: 25.5mA

Figure 62:
LED3 CURRENT CONTROL Register

Register: 0x28		LED3 CURRENT CONTROL		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:0	LED3_CURRENT	1010 1111	R/W	This register controls the output current of current source LED3 in 100µA steps from 0µA up to 25.5mA. 0000 0000: 0mA 0000 0001: 0.1mA ... **1010 1111: 17.5mA** ... 1111 1110: 25.4mA 1111 1111: 25.5mA

Figure 63:
LED4 CURRENT CONTROL Register

Register: 0x29		LED4 CURRENT CONTROL		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:0	LED4_CURRENT	1010 1111	R/W	This register controls the output current of current source LED4 in 100µA steps from 0µA up to 25.5mA. 0000 0000: 0mA 0000 0001: 0.1mA ... **1010 1111: 17.5mA** ... 1111 1110: 25.4mA 1111 1111: 25.5mA

Figure 64:
LED5 CURRENT CONTROL Register

Register: 0x2A		LED5 CURRENT CONTROL		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:0	LED5_CURRENT	1010 1111	R/W	This register controls the output current of current source LED5 in 100µA steps from 0µA up to 25.5mA. 0000 0000: 0mA 0000 0001: 0.1mA ... **1010 1111: 17.5mA** ... 1111 1110: 25.4mA 1111 1111: 25.5mA

Figure 65:
LED6 CURRENT CONTROL Register

Register: 0x2B		LED6 CURRENT CONTROL		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:0	LED6_CURRENT	1010 1111	R/W	This register controls the output current of current source LED6 in 100µA steps from 0µA up to 25.5mA. 0000 0000: 0mA 0000 0001: 0.1mA ... **1010 1111: 17.5mA** ... 1111 1110: 25.4mA 1111 1111: 25.5mA

Figure 66:
LED7 CURRENT CONTROL Register

Register: 0x2C		LED7 CURRENT CONTROL		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:0	LED7_CURRENT	1010 1111	R/W	This register controls the output current of current source LED7 in 100µA steps from 0µA up to 25.5mA. 0000 0000: 0mA 0000 0001: 0.1mA ... **1010 1111: 17.5mA** ... 1111 1110: 25.4mA 1111 1111: 25.5mA

Figure 67:
LED8 CURRENT CONTROL Register

Register: 0x2D		LED8 CURRENT CONTROL		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:0	LED8_CURRENT	1010 1111	R/W	This register controls the output current of current source LED8 in 100µA steps from 0µA up to 25.5mA. 0000 0000: 0mA 0000 0001: 0.1mA ... **1010 1111: 17.5mA** ... 1111 1110: 25.4mA 1111 1111: 25.5mA

Figure 68:
LED9 CURRENT CONTROL Register

Register: 0x2E		LED9 CURRENT CONTROL		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:0	LED9_CURRENT	1010 1111	R/W	This register controls the output current of current source LED9 in 100µA steps from 0µA up to 25.5mA. 0000 0000: 0mA 0000 0001: 0.1mA ... **1010 1111: 17.5mA** ... 1111 1110: 25.4mA 1111 1111: 25.5mA

MISC

This register contains miscellaneous control bits like the clock detection. Program execution is clocked with internal 32.7 kHz clock or with external clock. External clock can be used if a clock signal is present on CLK-pin. The external clock frequency must be 32.7 kHz in oder to meet the timing specifications of the datasheet and for correct operation. If a higher or a lower frequency is used, it will affect on the program execution engine operation speed. The detector block does not limit the maximum frequency. External clock status can be checked with read only bit EXT_CLK_USED in register address 3A, when the external clock detection is enabled (Bit [1] CLK_DET_EN = high).

If external clock is not used in the application, CLK pin should be connected to GND to avoid oscillation on this pin and extra current consumption.

Figure 69:
OUTPUT ON/OFF CONTROL LSB Register

Register: 0x05		OUTPUT ON/OFF CONTROL LSB		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7	VARIABLE_D_SEL	0	R/W	The variable D can be linked to two different sources. The default source for variable D is register 0x3C but it can be assigned to the LED test ADC output. This allows, for example, program execution control with an analog signal. **0: variable D source is register 0x3C** 1: variable D source is LED test ADC.
6	EN_AUTO_INCR	1	R/W	The automatic increment feature of the serial bus address enables a quick memory write of successive registers within one transmission. 0: serial bus address automatic increment is disabled. **1: serial bus address automatic increment is enabled.**
5	POWERSAVE_EN	0	R/W	Please refer to POWER SAVE for a detailed description of the powersave mode of AS3661. **0: power save mode is disabled.** 1: power save mode is enabled.

Register: 0x05		OUTPUT ON/OFF CONTROL LSB		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
4:3	CP_MODE	00	R/W	This register bits control the operation mode of the integrated charge pump. The charge pump can be switched OFF, forced to bypass mode, forced to 1.5x mode and automatic operation. **00: CP is switched OFF.** 01: CP is forced to bypass mode (1x). 10: CP is forced to 1.5x mode (output voltage is 4.5V) 11: CP is in automatic mode depending on load conditions
2	PWM_PS_EN	0	R/W	For a detailed description of this power save mode please refer to section POWER SAVE. This mode can only be used if the CP is in OFF mode or 1x mode. **0: PWM power save mode disabled.** 1: PWM power save mode enabled.
1	CLK_DET_EN	00	R/W	The following bits define the clock selection of AS3661. **00: forced external clock (CLK pin).** 01: forced internal clock. 10: automatic clock selection. 11: internal clock.
0	INT_CLK_EN			

ENGINEx PC

The program counter defines the starting value for each program execution engine. It can be any value between 000 0000 to 101 1111. The maximum value depends on program memory allocation between the three program execution engines.

Figure 70:
ENGINE1 PC Register

Register: 0x37		ENGINE1 PC		
Bit	Bit Name	Default	Access	Bit Description
6:0	ENGINE1_PC	000 0000	R/W	Program counter value for execution engine1 from 000 0000 to 101 1111 depending on the memory allocation of the application.

Figure 71:
ENGINE2 PC Register

Register: 0x38		ENGINE2 PC		
Bit	Bit Name	Default	Access	Bit Description
6:0	ENGINE2_PC	000 0000	R/W	Program counter value for execution engine2 from 000 0000 to 101 1111 depending on the memory allocation of the application.

Figure 72:
ENGINE3 PC Register

Register: 0x39		ENGINE3 PC		
Bit	Bit Name	Default	Access	Bit Description
6:0	ENGINE3_PC	000 0000	R/W	Program counter value for execution engine3 from 000 0000 to 101 1111 depending on the memory allocation of the application.

STATUS/INTERRUPT

This register contains several status and interrupt registers.

Figure 73:
STATUS / INTERRUPT Register

Register: 0x3A		STATUS / INTERRUPT		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7	LEDTEST_MEAS_DONE	0	R/W	This bit indicates when the LED test is done, and the result is written to the LED_TEST_ADC register (0x42). Typically the conversion takes 2.7 milliseconds to complete. The bit will not be cleared after conversion. Each write command to this register starts another conversion. **0: LED test not done.** 1: LED test done.
6	MASK_BUSY	1	R/W	Mask bit for interrupts generated by STARTUP_BUSY or ENGINE_BUSY. 0: External interrupt will be generated when STARTUP_BUSY or ENGINE_BUSY condition is no longer true. Reading the register 3A clears the status bits [5:4] and releases INT pin to high state. **1: Interrupt events will be masked i.e. no external interrupt will be generated from STARTUP_BUSY or ENGINE_BUSY event (default).**
5	STARTUP_BUSY	0	R	A status bit which indicates that the device is running the internal start-up sequence. Please note that STARTUP_BUSY bit is always "1" when CHIP_EN bit is "0". Please refer to STARTUP for detailed startup mode description. **0: internal start-up sequence completed.** 1: internal start-up sequence running.
4	ENGINE_BUSY	0	R	A status bit which indicates that a program execution engine is clearing internal registers. Serial bus master should not write or read program memory, or registers 0x00, 0x37 to 0x39 or 0x4C to 0x4E, when this bit is set to "1". **0: engine ready.** 1: at least one of the engines is clearing internal registers.

Register: 0x3A		STATUS / INTERRUPT		
Bit	Bit Name	Default	Access	Bit Description
3	EXT_CLK_USED	0	R	This bit is high when external clock signal on CLK pin is detected. CLK_DET_EN bit high in address 36 enables the clock detection. **0: external clock not detected.** 1: external clock detected.
2	ENG1_INT	0	R	This is the interrupt status bit for program execution engine 1. The bit is set by END or INT instruction. Reading the interrupt bit clears the interrupt. **0: interrupt unset/cleared.** 1: interrupt set.
1	ENG2_INT	0	R	This is the interrupt status bit for program execution engine 2. The bit is set by END or INT instruction. Reading the interrupt bit clears the interrupt. **0: interrupt unset/cleared.** 1: interrupt set.
0	ENG3_INT	0	R	This is the interrupt status bit for program execution engine 3. The bit is set by END or INT instruction. Reading the interrupt bit clears the interrupt. **0: interrupt unset/cleared.** 1: interrupt set.

GPO

AS3661 has one General Purpose Output pin (GPO). Status of the pin can be controlled with this register. Also, INT pin can be configured to function as a GPO by setting the bit EN_GPO_INT. When INT is configured to function as a GPO, output level is defined by the VBAT voltage.

When INT pin's GPO function is disabled, it operates as an open drain pin. INT signal is active low, i.e. when interrupt signal is send, the pin is pulled to GND. External pull-up resistor is needed for proper functionality.

Figure 74:
GPO Register

Register: 0x3B		GPO		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
2	INT_CONF	0	R/W	This bit defines the function of GPO pin. It can either be configured as interrupt pin or as general purpose output pin. **0: INT pin is set to function as an interrupt pin.** 1: INT pin is configured to function as a GPO.
1	GPO	0	R/W	This register controls the state of pin GPO. **0: GPO pin state is low.** 1: GPO pin state is high. GPO pin is a digital CMOS output, and no pulldown resistor is needed.
0	INT_GPO	0	R/W	If INT pin is defined as general purpose output (INT_CONF bit must be set to"1"), it is possible to control the INT pin with this bit. **0: INT pin state is low (if INT_CONF = 1).** 1: INT pin state is high (if INT_CONF = 1).

VARIABLE

The variable can be sued to store data in order to control for example the data flow.

Figure 75:
GPO Register

Register: 0x3C		GPO		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:0	VARIABLE_D	0000 0000	R/W	These bits are used for storing a global 8–bit variable. Variable can be used to control program flow.

RESET

Figure 76:
RESET Register

Register: 0x3D		RESET		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:0	RESET	0000 0011	R/W	Writing 11111111 into this register resets the AS3661. Internal registers are reset to the default values. Reading RESET register returns 00000011.

TEMP ADC CONTROL

Figure 77:
TEMP ADC CONTROL Register

Register: 0x3E		TEMP ADC CONTROL		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7	TEMP_MEAS_BUSY	0	R	Indicates the status of the temperature measurement ADC of AS3661. **0: temperature measurements done or not activated.** 1: temperature measurement active.
2	EN_TEMP_SENSOR	0	R/W	Every time when EN_TEMP_SENSOR is written high a new measurement period is started. The length of the measurement period depends on temperature. At 25°C a measurement takes 20 milliseconds. Temperature can be read from register 0x3F. **0: temperature sensor disabled.** 1: enable internal temperature sensor and start measurement.
1	CONTINUOUS_CONV	0	R/W	When EN_TEMP_SENSOR bit is set to"1" it is possible to enable a continuous temperature conversation setting the CONTINUOUS_CONV bit in this register. **0: new temperature measurement period initiated during start-up or after exit from power save mode.** 1: continuous temperature measurement. Not active when the device is in powersave.
0	SEL_EXT_TEMP	0	R/W	It is possible to link the temperature compensation register either to the internal temperature measurement result register 0x3F or to the TEMPERATURE WRITE register 0x40. This register can be sued to store the temperature of an external temperature measurement device to AS3661 in order to use it for LED temperature compensation. **0: temperature compensation source register addr 3FH.** 1: temperature compensation source register addr 40H.

TEMPERATURE READ

Figure 78:
TEMPERATURE READ Register

Register: 0x3F		TEMPERATURE READ		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:0	TEMPERATURE _READ	0001 1001	R	These bits are used for storing an 8-bit temperature reading acquired from the internal temperature sensor. This register is a read-only register. Temperature reading is stored in 8-bit two's complement format, see the table below. 1101 1010: -38°C ... **0001 1001: 25°C** ... 0101 1000: 88°C 0101 1001: 89°C

Note(s): When writing temperature data outside the range of the temperature compensation: Values greater than 89°C will be set to 89°C; values less than -38°C will be set to -38°C.

TEMPERATURE WRITE

Figure 79:
TEMPERATURE WRITE Register

Register: 0x40		TEMPERATURE WRITE		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:0	TEMPERATURE _WRITE	0000 0000	R/W	These bits are used for storing an 8-bit temperature reading acquired from an external temperature sensor, if such a sensor is used. Temperature reading is stored in 8-bit two's complement format, see the table below. 1101 1010: -38°C ... **0001 1001: 25°C** ... 0101 1000: 88°C 0101 1001: 89°C

Note(s): When writing temperature data outside the range of the temperature compensation: Values greater than 89°C will be set to 89°C; values less than -38°C will be set to -38°C.

LED TEST CONTROL

Figure 80:
LED TEST CONTROL Register

Register: 0x41		LED TEST CONTROL		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7	EN_LED_TEST_ADC	0	R/W	Writing this bit high (1) fires single LED test conversation. Thus each time you want to start a conversion it is necessary to write a "1" to this register. The measurement cycle is 2.7 milliseconds per conversion. **0: LED test measurement disabled.** 1: LED test measurement enabled
6	EN_LED_TEST_INT	0	R/W	This register enabled the interrupt for the LED test ADC. Interrupt can be cleared by reading STATUS/INTERRUPT register 0x3A. **0: no interrupt signal will be send to the INT pin when the LED test is accomplished.** 1: interrupt signal will be send to the INT pin when the LED test is accomplished.
5	CONTINUOUS _CONV	0	R/W	When EN_LED_TEST_ADC bit is set to"1", it is possible to enable a continuous conversation setting the CONTINUOUS_CONV bit to "1" in this register. **0: continuous conversion is disabled.** 1: continuous LED test measurement. Not active in powersave mode.

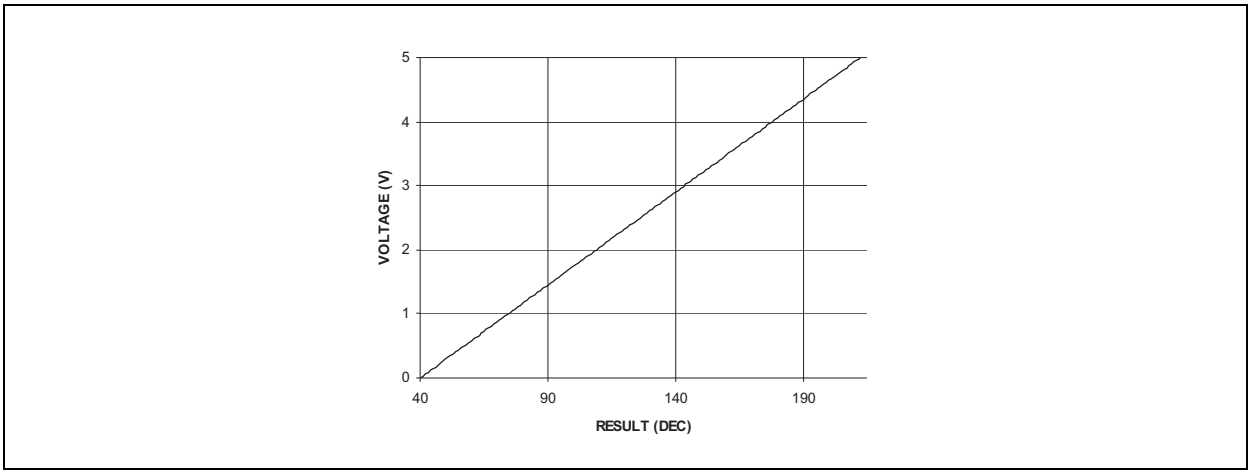
Register: 0x41		LED TEST CONTROL		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
4:0	LED_TEST_CTRL	0 0000	R/W	These bits are used for choosing the LED driver output to be measured with the LED test ADC. In addition to the LED outputs is is possible to measure VDD , INT-pin and charge-pump output voltage as well. **0 0000: LED1** 0 0001: LED2 0 0010: LED3 0 0011: LED4 0 0100: LED5 0 0101: LED6 0 0110: LED7 0 0111: LED8 0 1000: LED9 0 1001 to 0 1110: reserved, do not use 0 1111: VCP 1 0000: VBAT 1 0001: INT-pin 10010 to 11111: reserved, do not use

LED TEST ADC

Figure 81:
LED TEST ADC Register

Register: 0x42		LED TEST ADC		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:0	LED_TEST_ADC	N/A	R	This is used to store the LED test result. Read-only register. LED test ADC least significant bit corresponds to 30mV. The measured voltage V (typ.) is calculated as follows: V = (RESULT(DEC) × 0.03 - 1.478 V. For example, if the result is 10100110 = 166(DEC), the measured voltage is 3.50V (typ.) (see Figure 82 on page 82).

Figure 82:
LED Test Results vs. Measured Voltage



ENGINE1 VARIABLE A

Figure 83:
ENGINE1 VARIABLE A Register

Register: 0x45		ENGINE1 VARIABLE A		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:0	ENGINE1_ VARIABLE_A	0000 0000	R	These bits are used for engine 1 as a local variable. The register is a read only register and can be used for example for arithmetic operations with the program execution engine.

ENGINE2 VARIABLE A

Figure 84:
ENGINE2 VARIABLE A Register

Register: 0x46		ENGINE2 VARIABLE A		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:0	ENGINE2_ VARIABLE_A	0000 0000	R	These bits are used for engine 2 as a local variable. The register is a read only register and can be used for example for arithmetic operations with the program execution engine.

ENGINE3 VARIABLE A

Figure 85:
ENGINE3 VARIABLE A Register

Register: 0x47		ENGINE3 VARIABLE A		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:0	ENGINE3_ VARIABLE_A	0000 0000	R	These bits are used for engine 3 as a local variable. The register is a read only register and can be used for example for arithmetic operations with the program execution engine.

MASTER FADER1

Figure 86:
MASTER FADER1 Register

Register: 0x48		MASTER FADER1		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:0	MASTER_FADER1	0000 0000	R	An 8-bit register to control all the LED-drivers mapped to MASTER FADER1. Master fader allows the user to control dimming of multiple LEDS with a single serial bus write. This is a faster method to control the dimming of multiple LEDs compared to the dimming done with the PWM registers (address 0x16 to 0x1E), which would need multiple writes.

MASTER FADER2

Figure 87:
MASTER FADER2 Register

Register: 0x49		MASTER FADER2		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:0	MASTER_FADER2	0000 0000	R	An 8-bit register to control all the LED-drivers mapped to MASTER FADER2. Master fader allows the user to control dimming of multiple LEDS with a single serial bus write. This is a faster method to control the dimming of multiple LEDs compared to the dimming done with the PWM registers (address 0x16 to 0x1E), which would need multiple writes.

MASTER FADER3

Figure 88:
MASTER FADER3 Register

Register: 0x4A		MASTER FADER3		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:0	MASTER_FADER3	0000 0000	R	An 8-bit register to control all the LED-drivers mapped to MASTER FADER3. Master fader allows the user to control dimming of multiple LEDS with a single serial bus write. This is a faster method to control the dimming of multiple LEDs compared to the dimming done with the PWM registers (address 0x16 to 0x1E), which would need multiple writes.

ENG1 PROG START ADDR

Figure 89:
ENG1 PROG START ADDR Register

Register: 0x4C		ENG1 PROG START ADDR		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
6:0	ENG1_PROG_ START_ADDR	0000 0000	R/W	The program memory start address for program execution engine 1 is defined in this register.

ENG2 PROG START ADDR

Figure 90:
ENG2 PROG START ADDR Register

Register: 0x4D		ENG2 PROG START ADDR		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
6:0	ENG2_PROG_ START_ADDR	0000 0000	R/W	The program memory start address for program execution engine 2 is defined in this register.

ENG3 PROG START ADDR

Figure 91:
ENG3 PROG START ADDR Register

Register: 0x4E		ENG3 PROG START ADDR		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
6:0	ENG3_PROG_ START_ADDR	0000 0000	R/W	The program memory start address for program execution engine 3 is defined in this register.

PROG MEM PAGE SELECT

Figure 92:
PROG MEM PAGE SEL Register

Register: 0x4F		PROG MEM PAGE SEL		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
2:0	PAGE_SEL	000	R/W	These bits select the program memory page. The program memory is divided into six pages of 16 instructions; thus the total amount of the program memory is 96 instructions. **000: Program Memory 0x00 - 0x0F selected.** 001: Program Memory 0x10 - 0x1F selected. 010: Program Memory 0x20 - 0x2F selected. 011: Program Memory 0x30 - 0x3F selected. 100: Program Memory 0x40 - 0x4F selected. 101: Program Memory 0x50 - 0x5F selected.

ENG1 MAPPING MSB

Figure 93:
ENG1 MAPPING MSB Register

Register: 0x70		ENG1 MAPPING MSB		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7	ENG1_GPO	0	R	**0: GPO pin non-mapped to the program exec. engine 1.** 1: GPO pin is mapped to the program execution engine 1.
0	ENG1_LED9	0	R	**0: LED9 pin non-mapped to the program exec. engine1.** 1: LED9 pin is mapped to the program execution engine 1.

ENG1 MAPPING LSB

Figure 94:
ENG1 MAPPING LSB Register

Register: 0x71		ENG1 MAPPING LSB		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7	ENG1_LED8	0	R	**0: LED8 pin non-mapped to the program exec. engine 1.** 1: LED8 pin is mapped to the program execution engine 1.
6	ENG1_LED7	0	R	**0: LED7 pin non-mapped to the program exec. engine 1.** 1: LED7 pin is mapped to the program execution engine 1.
5	ENG1_LED6	0	R	**0: LED6 pin non-mapped to the program exec. engine 1.** 1: LED6 pin is mapped to the program execution engine1.
4	ENG1_LED5	0	R	**0: LED5 pin non-mapped to the program exec. engine 1.** 1: LED5 pin is mapped to the program execution engine 1.
3	ENG1_LED4	0	R	**0: LED4 pin non-mapped to the program exec. engine 1.** 1: LED4 pin is mapped to the program execution engine 1.
2	ENG1_LED3	0	R	**0: LED3 pin non-mapped to the program exec. engine 1.** 1: LED3 pin is mapped to the program execution engine 1.
1	ENG1_LED2	0	R	**0: LED2 pin non-mapped to the program exec. engine 1.** 1: LED2 pin is mapped to the program execution engine 1.
0	ENG1_LED1	0	R	**0: LED1 pin non-mapped to the program exec. engine 1.** 1: LED1 pin is mapped to the program execution engine 1.

ENG2 MAPPING MSB

Figure 95:
ENG2 MAPPING MSB Register

Register: 0x72		ENG2 MAPPING MSB		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7	ENG2_GPO	0	R	**0: GPO pin non-mapped to the program exec. engine 2.** 1: GPO pin is mapped to the program execution engine 2.
0	ENG2_LED9	0	R	**0: LED9 pin non-mapped to the program exec. engine 2.** 1: LED9 pin is mapped to the program execution engine 2.

ENG2 MAPPING LSB

Figure 96:
ENG2 MAPPING LSB Register

Register: 0x73		ENG2 MAPPING LSB		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7	ENG2_LED8	0	R	**0: LED8 pin non-mapped to the program exec. engine 2.** 1: LED8 pin is mapped to the program execution engine 2.
6	ENG2_LED7	0	R	**0: LED7 pin non-mapped to the program exec. engine 2.** 1: LED7 pin is mapped to the program execution engine 2.
5	ENG2_LED6	0	R	**0: LED6 pin non-mapped to the program exec. engine 2.** 1: LED6 pin is mapped to the program execution engine 2.
4	ENG2_LED5	0	R	**0: LED5 pin non-mapped to the program exec. engine 2.** 1: LED5 pin is mapped to the program execution engine 2.
3	ENG2_LED4	0	R	**0: LED4 pin non-mapped to the program exec. engine 2.** 1: LED4 pin is mapped to the program execution engine 2.
2	ENG2_LED3	0	R	**0: LED3 pin non-mapped to the program exec. engine 2.** 1: LED3 pin is mapped to the program execution engine 2.
1	ENG2_LED2	0	R	**0: LED2 pin non-mapped to the program exec. engine 2.** 1: LED2 pin is mapped to the program execution engine 2.
0	ENG2_LED1	0	R	**0: LED1 pin non-mapped to the program exec. engine 2.** 1: LED1 pin is mapped to the program execution engine 2.

ENG3 MAPPING MSB

Figure 97:
ENG3 MAPPING MSB Register

Register: 0x74		ENG3 MAPPING MSB		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7	ENG3_GPO	0	R	**0: GPO pin non-mapped to the program exec. engine 3.** 1: GPO pin is mapped to the program execution engine 3.
0	ENG3_LED9	0	R	**0: LED9 pin non-mapped to the program exec. engine 3.** 1: LED9 pin is mapped to the program execution engine 3.

ENG3 MAPPING LSB

Figure 98:
ENG3 MAPPING LSB Register

Register: 0x75		ENG3 MAPPING LSB		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7	ENG3_LED8	0	R	**0: LED8 pin non-mapped to the program exec. engine 3.** 1: LED8 pin is mapped to the program execution engine 3.
6	ENG3_LED7	0	R	**0: LED7 pin non-mapped to the program exec. engine 3.** 1: LED7 pin is mapped to the program execution engine 3.
5	ENG3_LED6	0	R	**0: LED6 pin non-mapped to the program exec. engine 3.** 1: LED6 pin is mapped to the program execution engine 3.
4	ENG3_LED5	0	R	**0: LED5 pin non-mapped to the program exec. engine 3.** 1: LED5 pin is mapped to the program execution engine 3.
3	ENG3_LED4	0	R	**0: LED4 pin non-mapped to the program exec. engine 3.** 1: LED4 pin is mapped to the program execution engine 3.
2	ENG3_LED3	0	R	**0: LED3 pin non-mapped to the program exec. engine 3.** 1: LED3 pin is mapped to the program execution engine 3.
1	ENG3_LED2	0	R	**0: LED2 pin non-mapped to the program exec. engine 3.** 1: LED2 pin is mapped to the program execution engine 3.
0	ENG3_LED1	0	R	**0: LED1 pin non-mapped to the program exec. engine 3.** 1: LED1 pin is mapped to the program execution engine 3.

GAIN CHANGE CTRL

With hysteresis and timer bits the user can optimize the charge pump performance to better meet the requirements of the application at hand. Some applications need to be optimized for efficiency and others need to be optimized for minimum EMI, for example.

Figure 99:
GAIN CHANGE CTRL Register

Register: 0x76		GAIN CHANGE CTRL		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7:6	TRESHOLD	00	R/W	Bits set the threshold voltage at which the charge pump gain changes from 1.5x to 1x. The threshold voltage is defined as the voltage difference between highest voltage output (LED1 to LED6) and input voltage V_{BAT}: $V_{TRESHOLD}$ = V_{BAT} -MAX (voltage on LED1 to LED6). If $V_{TRESHOLD}$ is larger than the set value (100mV to 400mV), the charge pump is in 1x mode. **00: 400mV** 01: 300mV 10: 200mV 11: 100mV
5	ADAPTIVE_ TRESH_EN	0	R/W	Gain change hysteresis prevents the mode from toggling back and forth (1x -> 1.5x -> 1x...) , which would cause ripple on V_{IN} and LED flicker. When the adaptive threshold is enabled, the width of the hysteresis region depends on the choice of TRESHOLD bits (see above), saturation of the current sources, charge pump load current, PWM overlap and temperature. **0: Adaptive threshold disabled.** 1: Adaptive threshold enabled.
4:3	TIMER	00	R/W	A forced mode change from 1.5x to 1.0x is attempted at the interval specified with these bits. Mode change is allowed if there is enough voltage over the LED drivers to ensure proper operation. Set FORCE_1x to "1" (see below) to activate this feature. **00: 5ms** 01: 10ms 10: 50ms 11: infinite

Register: 0x76		GAIN CHANGE CTRL		
Bit	Bit Name	Default	Access	Bit Description
2	FORCE_1x	0	R	Activates forced mode change. In forced mode charge pump mode change from 1.5x to 1x is attempted at the interval specified with the TIMER bits. **0: forced mode changes disabled.** 1: forced mode changes disabled.

Note(s): Values above are typical and should not be used as product specification. Writing to TRESHOLD [7:6] bits by the user overrides factory settings. Factory settings aren't user accessible.

Instruction Set

AS3661 has three independent programmable execution engines. All the program execution engines have their own program memory block allocated by the user. Note that in order to access program memory the operation mode needs to be load program, at least for one of the three program execution engines. Program execution is clocked with 32 768Hz clock. This clock can be generated internally or external 32 kHz clock can be connected to CLK32K pin. Using external clock enables synchronization of LED timing to the external clock signal.

Supported instruction set is listed in the tables below:

Figure 100:
LED Driver Instructions

LED Driver Instructions	Compiler Command	Bit [15]	Bit [14]	Bit [13]	Bit [12]	Bit [11]	Bit [10]	Bit [9]	Bit [8]	Bit [7]	Bit [6]	Bit [5]
ramp (1)	RMP	0	pre-scale	step time					sign			
ramp (2)	RWV	1	0	0	0	0	1	0	0	0	0	pre-scal
set_pwm (1)	SPW	0	1	0	0	0	0	0	0			
set_pwm (2)	SPV	1	0	0	0	0	1	0	0	0	1	1
wait	WAIT	0	pre-scale	time					0	0	0	0

Note(s) and/or Footnote(s):

1. This opcode is used with numerical operands.
2. This opcode is used with variables.

Figure 101:
LED Mapping Instructions

LED Mapping Instructions	Compiler Command	Bit [15]	Bit [14]	Bit [13]	Bit [12]	Bit [11]	Bit [10]	Bit [9]	Bit [8]	Bit [7]	Bit [6]	Bit [5]
mux_ld_start	MLS	1	0	0	1	1	1	1	0	0		
mux_map_start	MMS	1	0	0	1	1	1	0	0	0		
mux_ld_end	MLE	1	0	0	1	1	1	0	0	1		
mux_sel	MSL	1	0	0	1	1	1	0	1	0		

LED Mapping Instructions	Compiler Command	Bit [15]	Bit [14]	Bit [13]	Bit [12]	Bit [11]	Bit [10]	Bit [9]	Bit [8]	Bit [7]	Bit [6]	Bit [5]
mux_clr	MCL	1	0	0	1	1	1	0	1	0	0	0
mux_map_next	MMN	1	0	0	1	1	1	0	1	1	0	0
mux_map_prev	MMP	1	0	0	1	1	1	0	1	1	1	0
mux_ld_next	MLN	1	0	0	1	1	1	0	1	1	0	0
mux_ld_prev	MLP	1	0	0	1	1	1	0	1	1	1	0
mux_ld_addr	MLA	1	0	0	1	1	1	1	1	0		
mux_map_addr	MMA	1	0	0	1	1	1	1	1	1		

Figure 102:
Branch Instructions

Branch Instructions	Compiler Command	Bit [15]	Bit [14]	Bit [13]	Bit [12]	Bit [11]	Bit [10]	Bit [9]	Bit [8]	Bit [7]	Bit [6]	Bit [5]
rst	RST	0	0	0	0	0	0	0	0	0	0	0
branch (1)	BRN	1	0	1	loop count							
branch (2)	BRV	1	0	0	0	0	1	1	step nu			
Int	INT	1	1	0	0	0	1	0	0	0	0	0
end	END	1	1	0	Int	reset	0	0	0	0	0	0
trigger	TRG	1	1	1	wait for trigger							
					ext.trig	x	x	E3	E2	E1	ext.trig	X

Branch Instructions	Compiler Command	Bit [15]	Bit [14]	Bit [13]	Bit [12]	Bit [11]	Bit [10]	Bit [9]	Bit [8]	Bit [7]	Bit [6]	Bit [5]
jne	JNE	1	0	0	0	1	0	0	Number of instructions skipped if the operation ret			
jl	JL	1	0	0	0	1	0	1				
jge	JGE	1	0	0	0	1	1	0				
je	JE	1	0	0	0	1	1	1				

Note(s) and/or Footnote(s):

1. This opcode is used with numerical operands.
2. This opcode is used with variables.
3. X stands for 'don't care'.

Figure 103:
Data Transfer and Arithmetic Instructions

Arithmetic Instructions	Compiler Command	Bit [15]	Bit [14]	Bit [13]	Bit [12]	Bit [11]	Bit [10]	Bit [9]	Bit [8]	Bit [7]	Bit [6]	Bit [5]
ld	LD	1	0	0	1	target variable		0	0			
add(1)	ADN	1	0	0	1			0	1			
add(2)	ADV	1	0	0	1			1	1	0	0	0
sub(1)	SBN	1	0	0	1			1	0			
sub(2)	SBV	1	0	0	1			1	1	0	0	0

Note(s) and/or Footnote(s):

1. This opcode is used with numerical operands..
2. This opcode is used with variables.

LED Driver Instructions

RAMP (Numerical Operands)

This is the instruction useful for smoothly changing from one PWM value into another PWM value on the LED1 to LED9 outputs, in other words generating ramps (with a negative or positive slope). AS3661 allows programming very fast and very slow ramps. Ramp instruction generates a PWM ramp, using the effective PWM value as a starting value. At each ramp step the output is incremented /decremented by one unit, unless the step time span is 0 or # of increments is 0. Time span for one ramp step is defined with prescale -bit [14] and step time -bits [13:9]. Prescale = 0 sets 0.49 ms cycle time and prescale = 1 sets 15.6 ms cycle time; so the minimum time span for one step is 0.49 ms (prescale * step time span = 0.49ms × 1) and the maximum time span is 15.6 ms × 31 = 484ms/step If all the step time bits [13:9] are set to zero, output value is incremented / decremented during one prescale on the whole. Number of increment's value defines how many steps will be taken during one ramp instruction: Increment maximum value is 255d, which corresponds increment from zero value to the maximum value. If PWM reaches minimum/maximum value (0/255) during the ramp instruction, ramp instruction will be executed to the end regardless of saturation. This enables ramp instruction to be used as a combined ramp & wait instruction. Ramp instruction is the wait instruction when the increment bits [7:0] are set to zero.

***COMPILER COMMAND SYNTAX*:** RMP, prescale[1], step time[4], sign[1], number of increments[8];

Figure 104:
RMP Parameter Description

Name	Value	Description
prescale	0	Divides master clock (32 768 Hz) by 16 = 2048 Hz -> 0.488 ms cycle time
	1	Divides master clock (32 768 Hz) by 512 = 64 Hz -> 15.625 ms cycle time
step time	0-31	One ramp increment done in (step time) × (prescale).
sign	0	Increase PWM output
	1	Decrease PWM output
# of increments	0-255	The number of increment/decrement cycles. **Note:** Value 0 takes the same time as increment by 1, but it is the wait instruction.

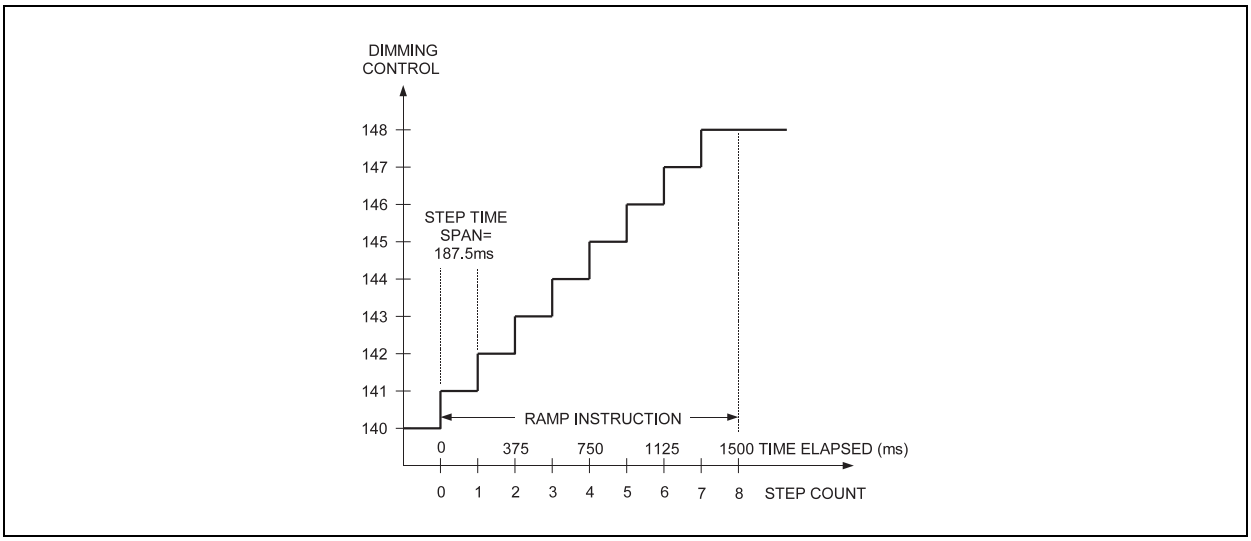
RMP Application Example

Let's say that the LED dimming is controlled according to the linear scale and effective PWM value at the moment t=0 is 140d (~55%,), as shown in the figure below, and we want to reach a PWM value of 148d (~58%) at the moment t = 1.5s. The parameters for the RAMP instruction will be:

- Prescale = 1 (15.625 ms cycle time).
- Step time = 12 (step time span will be 12*15.625 ms = 187.5 ms).
- Sign = 0 (increase PWM output).
- Number of increments = 8 (take 8 steps).

COMPILER COMMAND SYNTAX EXAMPLE: RMP, 1, 12, 0, 8;

Figure 105:
RAMP Instruction Example



RAMP (Variables)

Programming ramps with variables is very similar to programming ramps with numerical operands. The only difference is that step time and number of increments are captured from variable registers, when the instruction execution is started. If the variables are updated after starting the instruction execution, it will have no effect on instruction execution. Again, at each ramp step the output is incremented/decremented by one unless step time is 0 or increment is 0. Time span for one step is defined with prescale and step time bits. Step time is defined with variable A, B, C or D. Variables A, B and C are set with ld-instruction. Variable D is a global variable and can be set by writing the VARIABLE register (address 0x3C). LED TEST ADC register (address 0x42) can be used as a source for the variable D, as well.

Note(s): Variable A is the only local variable which can be read throughout the serial bus. Of course, the variable stored in 3CH can be read (and written), too. Setting register 0x06, 0x07, or

0x08 bit LOG_EN high/low sets logarithmic (1) or linear ramp (0). By using the logarithmic ramp setting the visual effect appears like a linear ramp, because the human eye behaves in a logarithmic way.

COMPILER COMMAND SYNTAX: RWV, prescale[1], sign[1], step time[2], number of increments[2];

Figure 106:
RWV Parameter Description

Name	Value	Description	
prescale	0	Divides master clock (32 768 Hz) by 16 = 2048 Hz -> 0.488 ms cycle time	
	1	Divides master clock (32 768 Hz) by 512 = 64 Hz -> 15.625 ms cycle time	
sign	0	Increase PWM output	
	1	Decrease PWM output	
step time	0-3	One ramp increment done in (step time) × (prescale). Step time is loaded with the value (5 LSB bits) of the variable defined below.	
		1	local variable A
		2	local variable B
		3	local variable C
		4	Register address 3CH variable D value, or register address 42H value.
		The value of the variable should be from 00001b to 11111b (1d to 31d) for correct operation.	
# of increments	0-3	The number of increment/decrement cycles. Value is taken from variable defined below:	
		0	local variable A
		1	local variable B
		2	local variable C
		3	Register address 0x3C variable D value, or register address 0x42 value.

RWV Application Example

Let's say that the LED dimming is controlled according to the linear scale and effective PWM value at the moment t=0 is 0d (0%,) and we want to reach a PWM value of 255d (100%) at the moment t = 3s. The parameters for the RAMP instruction will be:

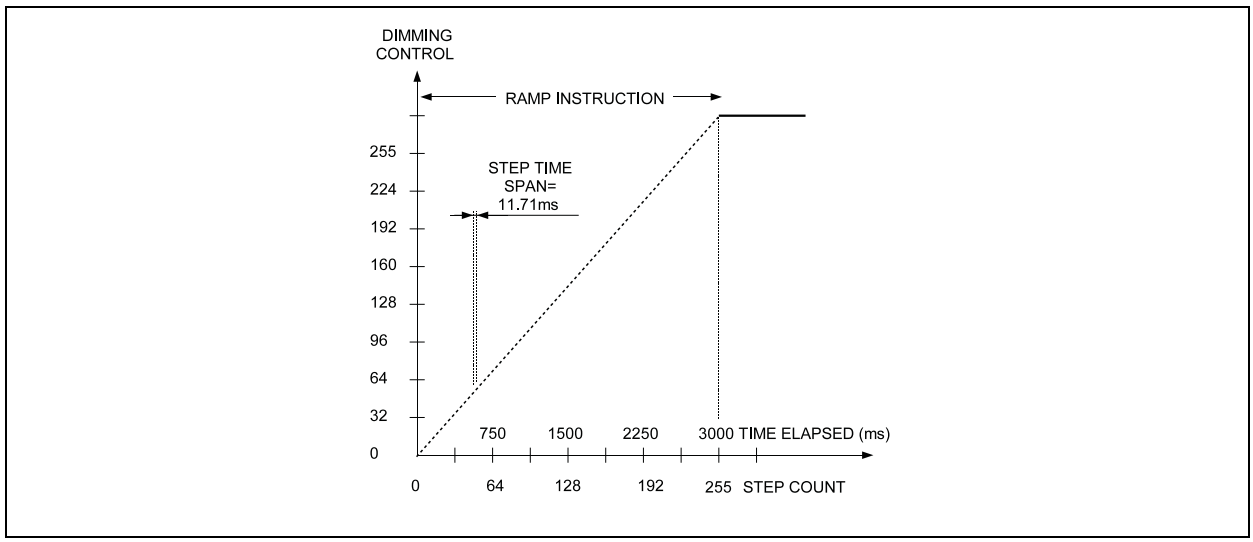
- Prescale = 0 (0.488 ms cycle time).
- Step time = 4 (use variable D in register 0x3C with a value of 24d).
- Sign = 0 (increase PWM output).

- Number of increments = 0 (use local variable A which must be loaded with the value 255d).

COMPILER COMMAND SYNTAX EXAMPLE: RMP, 0, 0, 4, 0;

The example above gives us a ramp time of 2.987s (tr = 0.488ms * 24 * 255).

Figure 107:
Ramp Instruction Example with Variables



SET PWM (Numerical Operands)

This instruction is used for setting the PWM value on the outputs LED1 to LED9 without any ramps. Set PWM output value from 0 to 255 with PWM value bits [7:0]. Instruction execution takes sixteen 32 kHz clock cycles (=488µs) .

COMPILER COMMAND SYNTAX: SPW, PWM Value[8];

Figure 108:
SPW Parameter Description

Name	Value	Description
PWM Value	0-255	PWM output duty cycle 0 - 100%

SPW Application Example

The SPW command can be used to set the PWM duty cycle of the program execution engine. In the following example we want to set the duty cycle of the PWM output to 55% like in the ramp example in the previous section. The right PWM value can be calculated with the following formula:

PWM value = (Duty Cycle * 255 / 100) = 55% * 255 / 100 = 140.

The predefined PWM value can be used as a starting point for dimming LEDs for example.

COMPILER COMMAND SYNTAX: SPW, 140;

SET PWM (Variables)

This instruction is used for setting the PWM value on the outputs LED1 to LED9 without any ramps. In comparison to the SPW command, this command is in combination with variables similar to the RWM example in one of the previous sections.

COMPILER COMMAND SYNTAX: SPV, Variable[2];

Figure 109:
SPW Parameter Description

Name	Value	Description	
Variable	0-3	0	local variable A
		1	local variable B
		2	global variable C
		3	Register address 3CH variable D value, or register address 42H value.

SPV Application Example

The purpose of the SPV command is basically the same one like with the SPW command in the previous section. The only difference is that this command allows the user the control the PWM duty cycle with the variables of the chip.

COMPILER COMMAND SYNTAX EXAMPLE: SPW, 0;

The example above shows the control of the duty cycle with the local variable A. If the local variable is for example loaded with a value of 100, the duty cycle of the PWM output is set to 39.2%.

WAIT

When a wait instruction is executed, the engine is set in a wait status and the PWM values on the outputs are frozen. This can be used for example to keep the LEDs enabled for a certain period of time before another up/down dimming process is being initiated.

COMPILER COMMAND SYNTAX: WAIT, prescale[1], time[5];

Figure 110:
WAIT Parameter Description

Name	Value	Description
Pre-scale	0	Divide master clock (32 768 Hz) by 16 which means 0.488 ms cycle time.
	1	Divide master clock (32 768 Hz) by 512 which means 15.625 ms cycle time.
time	1-31	Total wait time will be = (time) × (prescale). Maximum 484 ms, minimum 0.488 ms.

WAIT Application Example

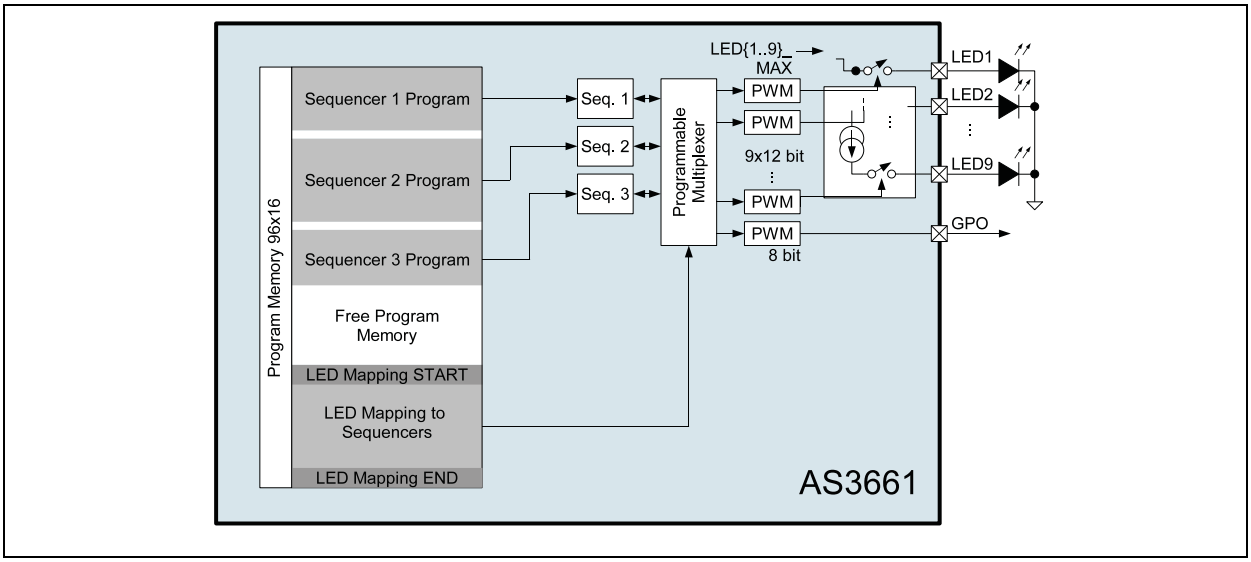
In the example shown below we want to have a target wait time of 125ms after dimming up the LEDs. In order to get the 100ms delay we select a prescaler value “1”, which gives a cycle time of 15.625ms. If we divide the 100ms by the cycle time we get the right value for the time parameter which is 8.

COMPILER COMMAND SYNTAX EXAMPLE: WAIT, 1, 8;

LED Mapping Instructions

These instructions define the engine-to-LED mapping. The mapping information is stored in a table, which is stored in the SRAM (program memory of the AS3661). AS3661 has three program execution engines which can be mapped to 9 LED drivers or to one GPO pin. One engine can control one or multiple LED drivers. The first part of the program memory of AS3661 is usually used for LED driver programs of each sequencer. The LED mapping is usually put at the end of the program memory where the programmable multiplexer, shown in the block diagram below, gets the information which LED must be connected to what sequencer output.

Figure 111:
LED Mapping Memory Allocation



In order to control and define the mapping of the LEDs there are totally eleven instructions for the engine-to-LED-driver control: mux_ld_start, mux_map_start, mux_ld_end, mux_sel, mux_clr, mux_map_next, mux_map_prev, mux_ld_next, mux_ld_prev, mux_ld_addr and mux_map_addr. With these instructions it is also possible to change the LED mapping from one mapping to another mapping, which has been defined in the LED mapping table, forth and back to create again more complex light patterns.

MUX_LD_START

Mux_ld_start defines the start of the mapping table location in the memory.

COMPILER COMMAND SYNTAX: MLS, SRAM address[7];

Figure 112:
MLS Parameter Description

Name	Value	Description
SRAM address	0-95	Mapping table start address

MLS Application Example

In this example we want to set the start address for the LED mapping table to 80d.

COMPILER COMMAND SYNTAX EXAMPLE: MLS, 80;

MUX_LD_END

Mux_ld_end defines the end of the mapping table location in the memory. It is very important to define the end address of the mapping table, otherwise it could happen if you use relative mapping commands, that the address pointer points to a position outside the mapping table due to the missing end address.

COMPILER COMMAND SYNTAX: MLE, SRAM address[7];

Figure 113:
MLE Parameter Description

Name	Value	Description
SRAM address	0-95	Mapping table end address

MLE Application Example

In this example we want to set the end address for the LED mapping table to 85d.

COMPILER COMMAND SYNTAX EXAMPLE: MLE, 85;

MUX_MAP_START

Mux_map_start defines the mapping table start address in the memory and the first row of the table will be activated (mapped) at the same time.

COMPILER COMMAND SYNTAX: MMP, SRAM address[7];

Figure 114:
MMP Parameter Description

Name	Value	Description
SRAM address	0-95	Mapping table start address

MMP Application Example

In the example we would like to set the start address to 80d. In addition to the definition of the start address of the mapping table the first LED mapping defined at address 80d gets activated. The difference to the MUSX_LD_START command, described in one of the previous sections, is that it only defines the start address without activating the LED mapping.

COMPILER COMMAND SYNTAX EXAMPLE: MMP, 80;

MUX_SEL

With mux_sel instruction one, and only one, LED driver (or the GPO-pin) can be connected to a program execution engine. Connecting multiple LEDs to one engine is done with the mapping table. After the mapping has been released from an LED, PWM register value will still control the LED brightness. If the mapping is released from the GPO pin, serial bus control takes over the GPO state.

COMPILER COMMAND SYNTAX: MSL, LED Select[6]

Figure 115:
MSL Parameter Description

Name	Value	Description	
LED Select	0-16	0	No drivers selected
		1	LED1 selected
		2	LED2 selected
		...	...
		16	GPO

MSL Application Example

In this example we would like to use the MSL command to map a single LED to a execution engine. Usually we do this with the mapping table but in case we want to use only a single LED on one sequencer it is possible to use the MSL command. The example command below shows the mapping of LED2 to the program execution engine.

COMPILER COMMAND SYNTAX EXAMPLE: MSL, 2;

MUX_CLR

Mux_clr clears engine-to-driver mapping. After the mapping has been released from an LED, PWM register value will still control the LED brightness. If the mapping is released from the GPO pin, serial bus control takes over the GPO state.

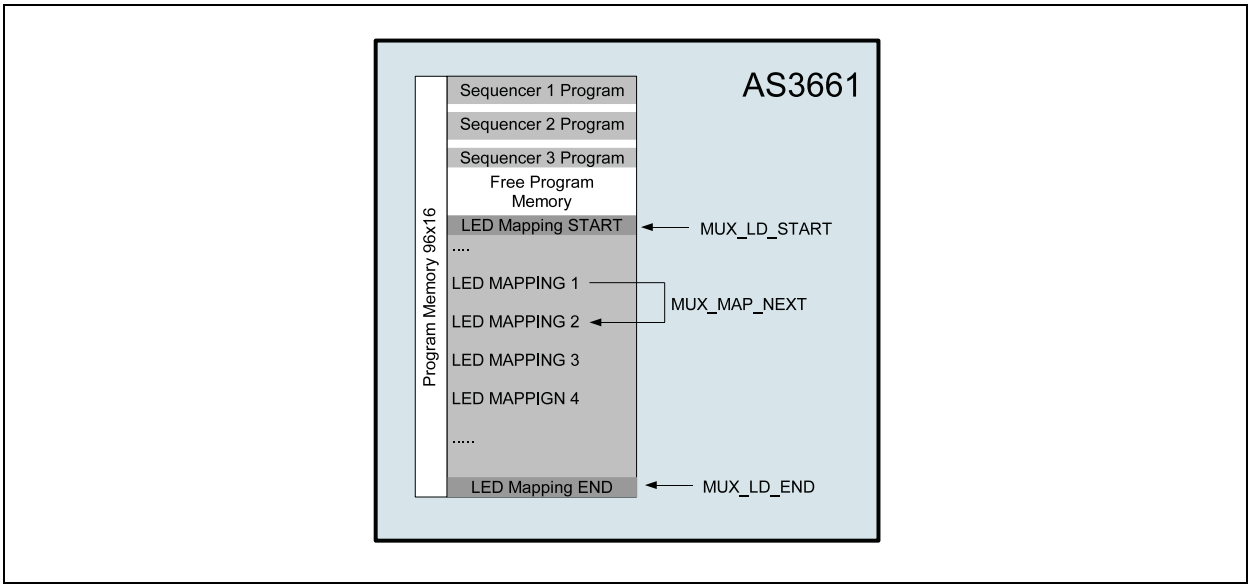
COMPILER COMMAND SYNTAX: MCL;

This command doesn't support any parameters.

MUX_MAP_NEXT

This instruction sets the next row active in the mapping table each time it is called. For example, if the 1st row is active at this moment, after mux_map_next instruction call the 2rd row will be active like in the block diagram shown in Figure 116 below.

Figure 116:
MUX_MAP_NEXT Command



If the mapping table end address is reached, activation will roll to the mapping table start address next time when the mux_map_next instruction is called. Engine will not push a new PWM value to the LED driver output before set_pwm or ramp instruction is executed. If the mapping has been released from an LED, the value in the PWM register will still control the LED brightness. If the mapping is released from the GPO pin, serial bus control takes over the GPO state.

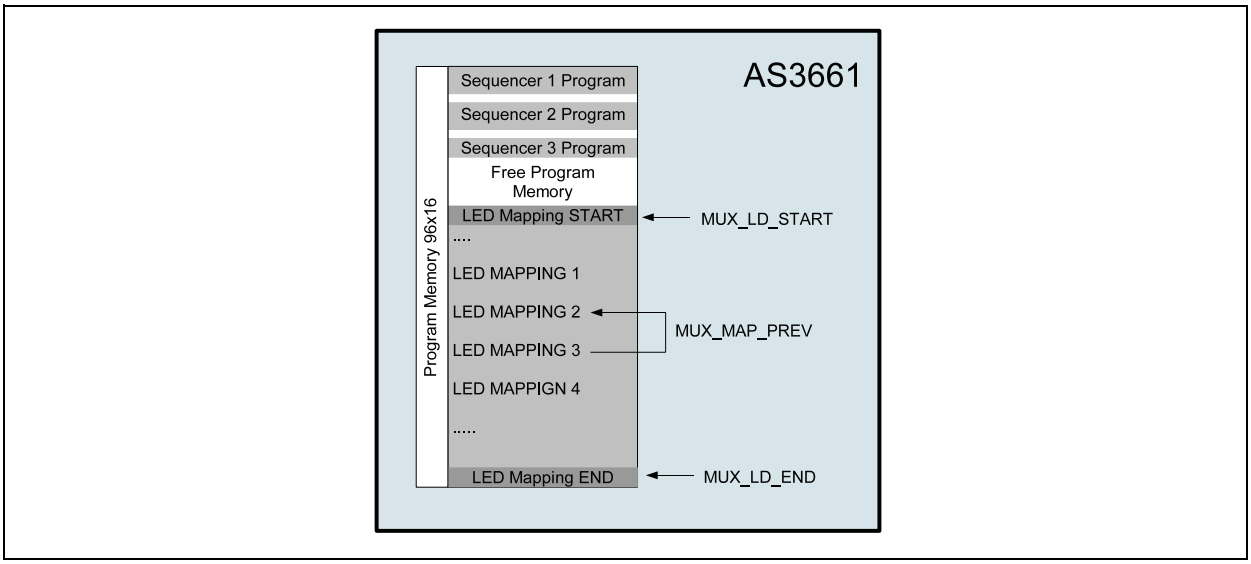
COMPILER COMMAND SYNTAX: MMN;

This command doesn't support any parameters.

MUX_MAP_PREV

This instruction sets the previous row active in the mapping table each time it is called. For example, if the 3rd row is active at this moment, after mux_map_prev instruction call the 2nd row will be active like in block diagram shown in Figure 117 below.

Figure 117:
MUX_MAP_PREV Command



If the mapping table start address is reached, activation will roll to the mapping table end address next time the mux_map_prev instruction is called. Engine will not push a new PWM value to the LED driver output before set_pwm or ramp instruction is executed. If the mapping has been released from an LED, the value in the PWM register will still control the LED brightness. If the mapping is released from the GPO pin, serial bus control takes over the GPO state.

COMPILER COMMAND SYNTAX: MMP;

This command doesn't support any parameters.

MUX_LD_NEXT

Similar than the mux_map_next instruction, but only the index pointer will be set to point to the next row i.e. no mapping will be set and the engine-to-LED-driver connection will not be updated.

COMPILER COMMAND SYNTAX: MLN;

This command doesn't support any parameters.

MUX_LD_PREV

Similar than the mux_map_prev instruction, but only the index pointer will be set to point to the previous row i.e. no mapping will be set and the engine-to-LED-driver connection will not be updated.

COMPILER COMMAND SYNTAX: MLP;

This command doesn't support any parameters.

MUX_MAP_ADDR

Mux_map_addr sets the index pointer to point the mapping table row defined by bits [6:0] and sets the row active. Engine will not push a new PWM value to the LED driver output before set_pwm or ramp instruction is executed. If the mapping has

been released from an LED, the value in the PWM register will still control the LED brightness. If the mapping is released from the GPO pin, serial bus control takes over the GPO state.

COMPILER COMMAND SYNTAX: MMA, SRAM address[7];.

Figure 118:
MMA Parameter Description

Name	Value	Description
SRAM address	0-95	Any SRAM address containing mapping data.

MMA Application Example

In this example we asume the we have aleady defined the start and end address of the LED mapping table. Now we want to set address 83d within the address range of the map table active.

COMPILER COMMAND SYNTAX EXAMPLE: MMA, 83;

MUX_LD_ADDR

Mux_ld_addr sets the index pointer to point the mapping table row defined by bits [6:0], but the row will not be set active.

COMPILER COMMAND SYNTAX: MLA, SRAM address;

Figure 119:
MLA Parameter Description

Name	Value	Description
SRAM address	0-95	Any SRAM address containing mapping data.

Branch Instructions

RST

RST instruction resets Program Counter register (address 37H, 38H, or 39H) and continues executing the program from the program start address defined in 4C-4E. Instruction takes sixteen 32 kHz clock cycles.

COMPILER COMMAND SYNTAX: RST;

This command doesn't support any parameters.

Note(s): The default value for all program memory registers is 0000H, which is the RST instruction.

BRANCH (Numerical)

Branch instruction is mainly indented for repeating a portion of the program code several times. Branch instruction loads step number value to program counter. Loop count parameter defines how many times the instructions inside the loop are

repeated. AS3661 supports nested looping i.e. loop inside loop. The number of nested loops is not limited. Instruction takes sixteen 32 kHz clock cycles.

COMPILER COMMAND SYNTAX: BRN, loop count[6], step number[7];

Figure 120:
BRN Parameter Description

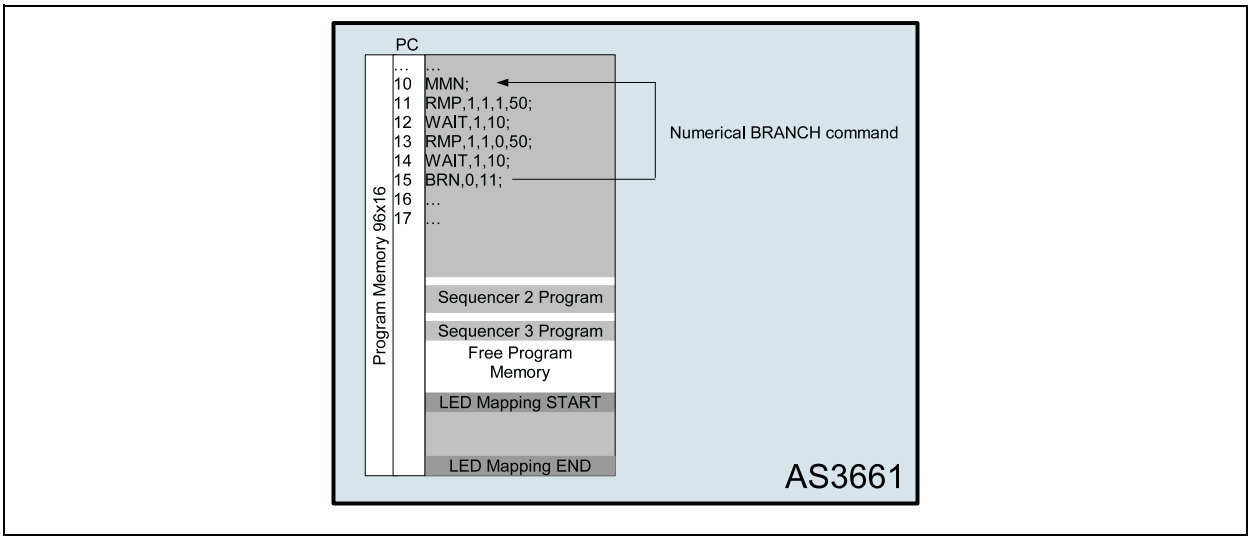
Name	Value	Description
loop count	0-63	The number of loops to be done. 0 means an infinite loop
step number	0-95	The step number to be loaded to program counter

BRN Application Example

In this application example we would like to create an infinite loop, which means the loop will never stop. The code we want to repeat has a start address of 10d. At the end of the code we want to repeat we put the BRN command with the program counter address 10d. Once the program execution engine executes the BRN command the program counter jumps back to address 10d and starts executing the code from this address until it reaches again the BRN command.

COMPILER COMMAND SYNTAX: EXAMPLE: BRN, 0, 10;

Figure 121:
BRN Example for Execution Engine 1



BRANCH (Variables)

The BRANCH command for variables has basically the same functionality like the numerical command. The only difference is that the loop count is controlled with variables instead of having a fixed number.

COMPILER COMMAND SYNTAX: BRV, step number[7], loop count[2];

Figure 122:
BRN Parameter Description

Name	Value	Description	
step number	0-95	The step number to be loaded to program counter	
loop count	0-3	Selects the variable for step number value. Step number is loaded with the value of the variable defined below	
		0	local variable A
		1	local variable B
		2	local variable C
		3	Register address 3CH variable D value, or register address 42H value

INT

Send interrupt to processor by pulling the INT pin down and setting corresponding status bit high. Interrupt can be cleared by reading interrupt bits in STATUS/INTERRUPT register at address 3A.

COMPILER COMMAND SYNTAX: INT;

This command doesn't support any parameters.

END

End program execution. Instruction takes sixteen 32 kHz clock cycles.

COMPILER COMMAND SYNTAX: END, int[1], Reset[1];

Figure 123:
END Parameter Description

Name	Value	Description
Int	0	No interrupt will be sent. PWM register values will remain intact.
	1	Reset program counter value to 0 and send interrupt to processor by pulling the INT pin down and setting corresponding status bit high to notify that program has ended. PWM register values will remain intact. Interrupt can be cleared by reading interrupt bits in STATUS/INTERRUPT register at address 3A.

Name	Value	Description
Reset	0	Reset program counter value to 0 and hold. PWM register values will remain intact.
	1	Reset program counter value to 0 and hold. PWM register values of the non-mapped drivers will remain. PWM register values of the mapped drivers will be set to "0000 0000". On completion of int instruction with this bit set to "1" the master fader registers are set to zero as follows: Program execution engine 1 sets MASTER FADER 1 (48H) to zero, engine 2 sets MASTER FADER 2 (49H) to zero and engine 3 sets MASTER FADER 3 (4AH) to zero.

END Application Example

The example code below sends an interrupt to the processor and resets the program counts to 0. The program execution engine is set on hold.

COMPILER COMMAND SYNTAX EXAMPLE: END, 1, 0;

TRIGGER

Wait or send triggers can be used to e.g. synchronize operation between the program execution engines. Send trigger instruction takes sixteen 32 kHz clock cycles and wait for trigger takes at least sixteen 32 kHz clock cycles. The receiving engine stores the triggers which have been sent. Received triggers are cleared by wait for trigger instruction. Wait for trigger instruction is executed until all the defined triggers have been received (note: several triggers can be defined in the same instruction).

External trigger input signal must stay low for at least two 32 kHz clock cycles to be executed. Trigger output signal is three 32 kHz clock cycles long. External trigger signal is active low, i.e. when trigger is send/received the pin is pulled to GND. Sent external trigger is masked, i.e. the device which has sent the trigger will not recognize it. If send and wait external trigger are used on the same instruction, the send external trigger is executed first, then the wait external trigger.

COMPILER COMMAND SYNTAX: TRG, wait for trigger[6], send a trigger[6]

Figure 124:
TRG Parameter Description

Name	Value	Description
wait for trigger	0-31	Wait for trigger from the engine(s). Several triggers can be defined in the same instruction. Bit [7] engages engine 1, bit [8] engine 2, bit [9] engine 3 and bit [12] is for external trigger I/O. Bits [10] and [11] are not in use.
send a trigger	0-31	Send a trigger to the engine(s). Several triggers can be defined in the same instruction. Bit [1] engages engine 1, bit [2] engine 2, bit [3] engine 3 and bit [6] is for external trigger I/O. Bits [4] and [5] are not in use.

TRG Application Example

In this example we want to wait/receive a trigger from program execution engine 1.

COMPILER COMMAND SYNTAX EXAMPLE: TRG, 2, 0;

JNE/JL/JGE/JE

AS3661 instruction set includes the following conditional jump instructions: jne (jump if not equal); jge (jump if greater or equal); jl (jump if less); je (jump if equal). If the condition is true a certain number of instructions will be skipped (i.e. the program jumps forward to a location relative to the present location). If condition is false then the next instruction will be executed.

COMPILER COMMAND SYNTAX: JNE, number of instructions...[5], Variable1[2], Variable2[2];

COMPILER COMMAND SYNTAX: JL, number of instructions...[5], Variable1[2], Variable2[2];

COMPILER COMMAND SYNTAX: JGE, number of instructions...[5], Variable1[2], Variable2[2];

COMPILER COMMAND SYNTAX: JE, number of instructions...[5], Variable1[2], Variable2[2];

Figure 125:
JNE/JL/JGE/JE Parameter Description

Name	Value	Description	
Number of instructions to be skipped if the operation returns true.	0-31	The number of instructions to be skipped when the statement is true. **Note:** Value 0 means redundant code.	
Variable1	0-3	Defines the variable to be used in the test:	
		0	local variable A
		1	local variable B
		2	global variable
		C3	Register address 3CH variable, or register address 42H value
Variable2	0-3	Defines the variable to be used in the test:	
		0	local variable A
		1	local variable B
		2	global variable
		C3	Register address 3CH variable, or register address 42H value

JNE Application Example

In the following example we compare local variable A with local variable B. If the value of the two registers is not equal the command will skip three instructions.

COMPILER COMMAND SYNTAX EXAMPLE: JNE, 3, 0,1;

Arithmetic Instructions

LD

This instruction is used to assign a value into a variable; the previous value in that variable is overwritten. Each of the engines have two local variables, called A and B. The variable C is a global variable which is shared with all three program execution engines.

COMPILER COMMAND SYNTAX: LD, Target Variable[2];

Figure 126:
LD Parameter Description

Name	Value	Description	
Target Variable	0-2	0	variable A
		1	variable B
		2	variable C
8-bit value	0-255	variable value	

LD Application Example

In this example we want to load variable B with a value of 100;

COMPILER COMMAND SYNTAX EXAMPLE: LD, 1, 100;

ADD (Numerical Operands)

Operator either adds the 8-bit value to the current value of the target variable.

COMPILER COMMAND SYNTAX: ADN, Target Variable,[2], 8-Bit value[8];

Figure 127:
ADN Parameter Description

Name	Value	Description	
8-bit value	0-255	Variable value	
Target Variable	0-2	0	variable A
		1	variable B
		2	variable C
Variable1	0-3	0	local variable A
		1	local variable B
		2	global variable
		C3	Register address 3CH variable, or register address 42H value
Variable2	0-3	0	local variable A
		1	local variable B
		2	global variable
		C3	Register address 3CH variable, or register address 42H value

ADN Application Example

In this example we would like to add a value of 100d to variable 'A', which is loaded with a value of 10d. The result of the operation is 110d stored in variable 'A'.

COMPILER COMMAND SYNTAX EXAMPLE: ADN, 0, 100;

ADD (Variables)

This command adds the value of the variable 1 (A, B, C or D) to the value of the variable 2 (A, B, C or D) and stores the result in the register of variable A, B or C which is defined as target variable. Variables overflow from 255 to 0.

COMPILER COMMAND SYNTAX: ADV, Target Variable[2], Variable1[2], Variable2[2];

Figure 128:
ADV Parameter Description

Name	Value	Description	
Target Variable	0-2	0	variable A
		1	variable B
		2	variable C
Variable1	0-3	0	local variable A
		1	local variable B
		2	global variable
		C3	Register address 3CH variable, or register address 42H value
Variable2	0-3	0	local variable A
		1	local variable B
		2	global variable
		C3	Register address 3CH variable, or register address 42H value

ADV Application Example

In this example we want to add variable 'A' to variable 'B'. The result should be stored in variable 'C'.

COMPILER COMMAND SYNTAX EXAMPLE: ADV, 2, 0, 1;

SUB (Numerical)

SUB Operator either subtracts the 8-bit value from the current value of the target variable.

COMPILER COMMAND SYNTAX: SBN, Target Variable[2], 8-bit value[8];

Figure 129:
SBN Parameter Description

Name	Value	Description	
8-bit value	0-255	variable value	
Target Variable	0-2	0	variable A
		1	variable B
		2	variable C

SBN Application Example

In this example we would like to subtract 50d from local variable 'A'. The result is stored in variable 'A'.

COMPILER COMMAND SYNTAX EXAMPLE: SBN, 0, 50;

SUB (Variables)

The SBV command subtracts the value of the variable 2 (A, B, C or D) from the value of the variable 1 (A, B, C or D) and stores the result in the register of target variable (A, B or C). Variables overflow from 0 to 255.

COMPILER COMMAND SYNTAX: SBV, Target Variable[2], Variable1[2], Variable2[2];

Figure 130:
SBV Parameter Description

Name	Value	Description	
Target Variable	0-2	0	variable A
		1	variable B
		2	variable C
Variable1	0-3	0	local variable A
		1	local variable B
		2	global variable
		C3	Register address 3CH variable, or register address 42H value
Variable2	0-3	0	local variable A
		1	local variable B
		2	global variable
		C3	Register address 3CH variable, or register address 42H value

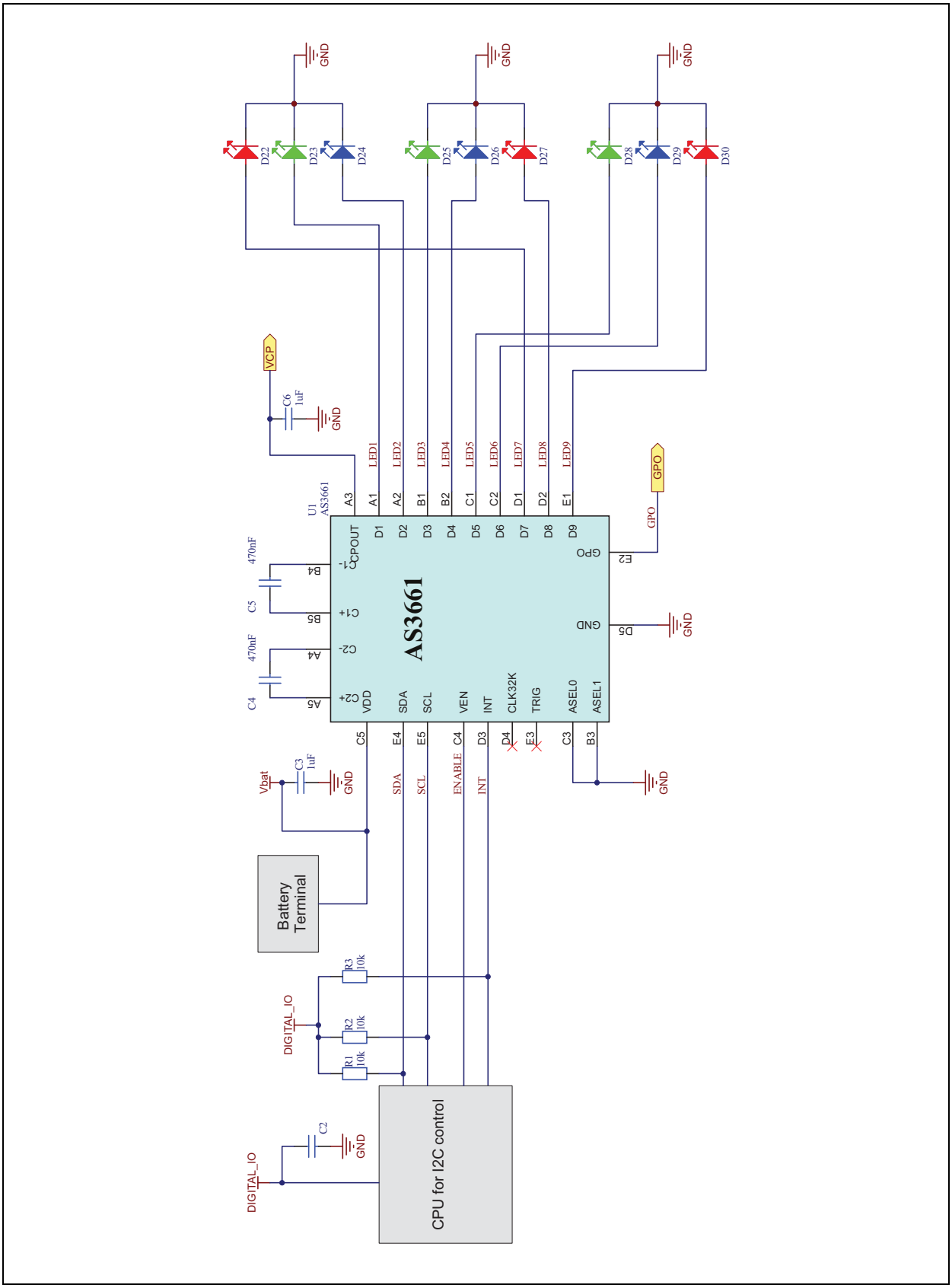
SBV Application Example

In this example we would like to subtract variable 'A' from variable 'B'. The result should be stored in variable 'C'.

COMPILER COMMAND SYNTAX EXAMPLE: SBV, 2, 0, 1;

Typical Application

Figure 131:
Typical Application 3 RGB LEDs



Recommended External Components

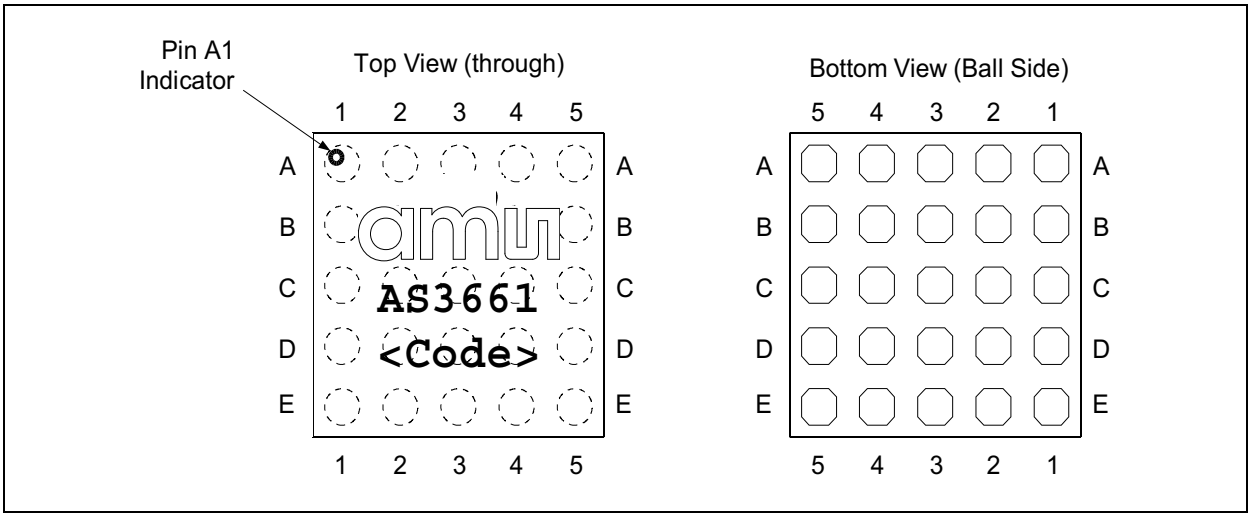
The AS3661 requires 4 external capacitors for proper operation. Surface-mount multi-layer ceramic capacitors are recommended. Tantalum and aluminium capacitors are not recommended because of their high ESR. For the flying capacitors (C1 and C2) multi-layer ceramic capacitors should always be used. These capacitors are small, inexpensive and have very low equivalent series resistance (ESR <20mΩ typ.). Ceramic capacitors with X7R or X5R temperature characteristic are preferred for use with the AS3661. These capacitors have tight capacitance tolerance (as good as ±10%) and hold their value over temperature (X7R: ±15% over -55°C to 125°C; X5R: ±15% over -55°C to 85°C). Capacitors with Y5V or Z5U temperature characteristic are generally not recommended for use with the AS3661. Capacitors with these temperature characteristics typically have wide capacitance tolerance (+80%, -20%) and vary significantly over temperature (Y5V: +22%, -82% over -30°C to 85°C range; Z5U: +22%, -56% over +10°C to 85°C range). Under some conditions, a nominal 1µF Y5V or Z5U capacitor could have a capacitance of only 0.1µF. Such detrimental deviation is likely to cause Y5V and Z5U capacitors to fail to meet the minimum capacitance requirements of the AS3661. For proper operation it is necessary to have at least 0.24µF of effective capacitance for each of the flying capacitors under all operating conditions. The output capacitor C_{VCPOUT} directly affects the magnitude of the output ripple voltage. In general, the higher the value of C_{VCPOUT}, the lower the output ripples magnitude. For proper operation it is recommended to have at least 0.50µF of effective capacitance for C_{VBAT} and C_{VCPOUT} under all operating conditions. The voltage rating of all four capacitors should be 6.3V; 10V is recommended. Recommended External Components below lists recommended external components from some leading ceramic capacitor manufacturers. It is strongly recommended that the AS3661 circuit be thoroughly evaluated early in the design-in process with the mass-production capacitors of choice. This will help ensure that any variability in capacitance does not negatively impact circuit performance.

Figure 132:
Recommended External Components

Model	Type	Vendor	Voltage Rating	Package Size
1µF for C_{VBAT} and C_{VCPOUT}				
C1005X5R1A105K	Ceramic X5R	TDK	10V	0402
LMK105BJ105KV-F	Ceramic X5R	Taiyo Yuden	10V	0402
ECJ0EB1A105M	Ceramic X5R	Panasonic	10V	0402
ECJUVBPA105	Ceramic X5R, array of two	Panasonic	10V	0504
470F for C_{FLY1} and C_{FLY2}				
C1005X5R1A474K	Ceramic X5R	TDK	10V	0402
LMK105BJ474KV-F	Ceramic X5R	Taiyo Yuden	10V	0402
ECJ0EB0J474K	Ceramic X5R	Panasonic	6.3V	0402
LEDs		User defined. Note that D7, D8 and D9 outputs are powered from V_{BAT} when specifying the LEDs.		

Package Drawings & Markings

Figure 133:
WL-CSP-25 (2.285 × 2.285mm) 0.4mm pitch Marking

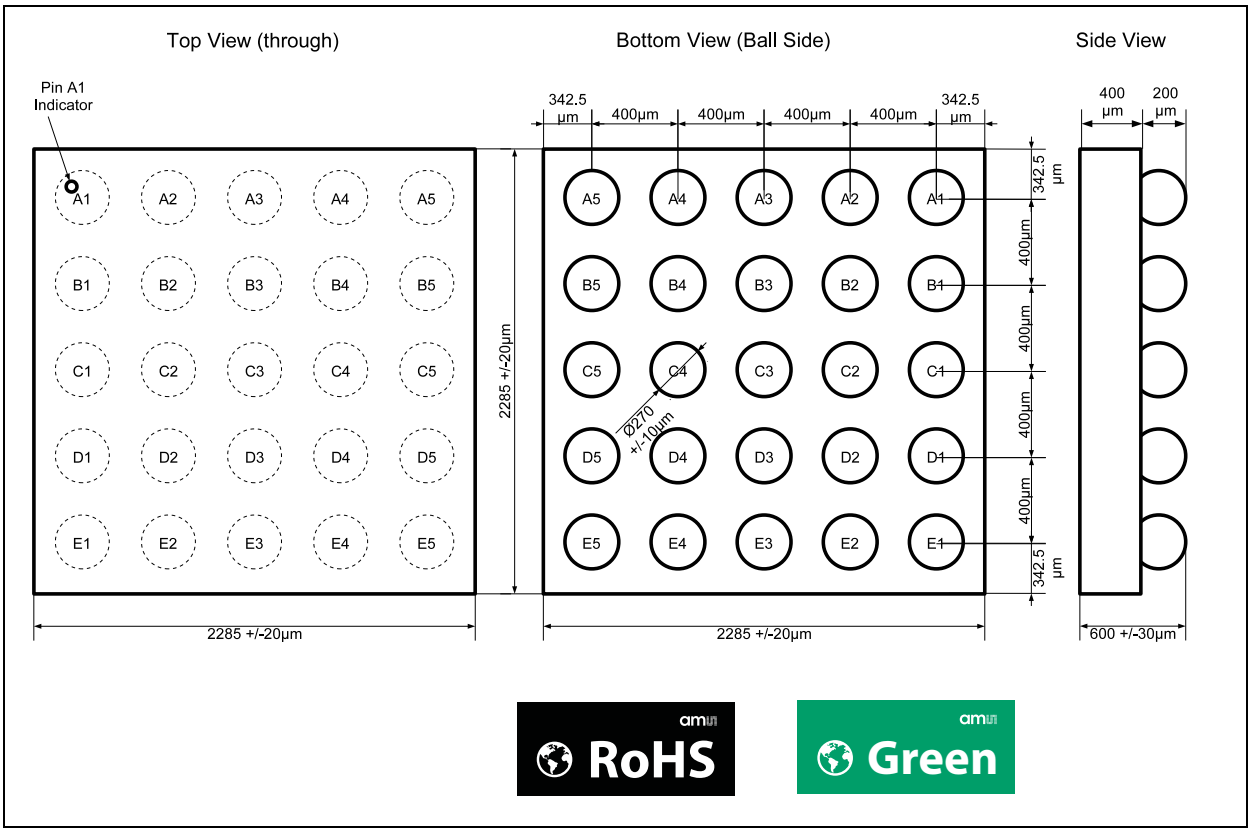


Line 1: ams logo

Line 2: AS3661

Line 3: <Code>
Tracecode (4 characters)

Figure 134:
WL-CSP-25 (2.285 × 2.285mm) 0.4mm pitch Package Dimensions



Ordering & Contact Information

The devices are available as the standard products shown in Figure 135.

Figure 135:
Ordering Information

Ordering Code	Marking	Description	Delivery Form	Package
AS3661-BWLT	AS3661	Programmable 9-channel LED Driver	Tape & Reel	WL-CSP-25 (2.285 × 2.285mm) 0.4mm pitch

Buy our products or get free samples online at:
www.ams.com/ICdirect

Technical Support is available at:
www.ams.com/Technical-Support

Provide feedback about this document at:
www.ams.com/Document-Feedback

For further information and requests, e-mail us at:
ams_sales@ams.com

For sales offices, distributors and representatives, please visit:
www.ams.com/contact

Headquarters
ams AG
Tobelbaderstrasse 30
8141 Unterpremstaetten
Austria, Europe

Tel: +43 (0) 3136 500 0

Website: www.ams.com

RoHS Compliant & ams Green Statement

RoHS: The term RoHS compliant means that ams AG products fully comply with current RoHS directives. Our semiconductor products do not contain any chemicals for all 6 substance categories, including the requirement that lead not exceed 0.1% by weight in homogeneous materials. Where designed to be soldered at high temperatures, RoHS compliant products are suitable for use in specified lead-free processes.

ams Green (RoHS compliant and no Sb/Br): ams Green defines that in addition to RoHS compliance, our products are free of Bromine (Br) and Antimony (Sb) based flame retardants (Br or Sb do not exceed 0.1% by weight in homogeneous material).

Important Information: The information provided in this statement represents ams AG knowledge and belief as of the date that it is provided. ams AG bases its knowledge and belief on information provided by third parties, and makes no representation or warranty as to the accuracy of such information. Efforts are underway to better integrate information from third parties. ams AG has taken and continues to take reasonable steps to provide representative and accurate information but may not have conducted destructive testing or chemical analysis on incoming materials and chemicals. ams AG and ams AG suppliers consider certain information to be proprietary, and thus CAS numbers and other limited information may not be available for release.

Copyrights & Disclaimer

Devices sold by ams AG are covered by the warranty and patent indemnification provisions appearing in its General Terms of Trade. ams AG makes no warranty, express, statutory, implied, or by description regarding the information set forth herein. ams AG reserves the right to change specifications and prices at any time and without notice. Therefore, prior to designing this product into a system, it is necessary to check with ams AG for current information. This product is intended for use in commercial applications. Applications requiring extended temperature range, unusual environmental requirements, or high reliability applications, such as military, medical life-support or life-sustaining equipment are specifically not recommended without additional processing by ams AG for each application. This product is provided by ams AG “AS IS” and any express or implied warranties, including, but not limited to the implied warranties of merchantability and fitness for a particular purpose are disclaimed.

ams AG shall not be liable to recipient or any third party for any damages, including but not limited to personal injury, property damage, loss of profits, loss of use, interruption of business or indirect, special, incidental or consequential damages, of any kind, in connection with or arising out of the furnishing, performance or use of the technical data herein. No obligation or liability to recipient or any third party shall arise or flow out of ams AG rendering of technical or other services.

Document Status

Document Status	Product Status	Definition
Product Preview	Pre-Development	Information in this datasheet is based on product ideas in the planning phase of development. All specifications are design goals without any warranty and are subject to change without notice
Preliminary Datasheet	Pre-Production	Information in this datasheet is based on products in the design, validation or qualification phase of development. The performance and parameters shown in this document are preliminary without any warranty and are subject to change without notice
Datasheet	Production	Information in this datasheet is based on products in ramp-up to full production or full production which conform to specifications in accordance with the terms of ams AG standard warranty as given in the General Terms of Trade
Datasheet (discontinued)	Discontinued	Information in this datasheet is based on products which conform to specifications in accordance with the terms of ams AG standard warranty as given in the General Terms of Trade, but these products have been superseded and should not be used for new designs

Revision Information

Changes from 1.3 to current revision 1-31 (2015-Feb-03)	Page
1.3 to 1-30 (2014-Jul-24)	
Content of austriamicrosystems datasheet was converted to latest **ams** design	
1-30 (2014-Jul-24) to 1-31 (2015-Feb-03)	
Added benefits to Figure 1	1

Note(s) and/or Footnote(s):

1. Page and figure numbers for the previous version may differ from page and figure numbers in the current revision.
2. Correction of typographical errors is not explicitly mentioned.

Content Guide

X-ON Electronics

Largest Supplier of Electrical and Electronic Components

Click to view similar products for LED Lighting Development Tools ***category:***

Click to view products by ams ***manufacturer:***

Other Similar products are found below :

MIC2870YFT EV ADP8860DBCP-EVALZ LM3404MREVAL ADM8843EB-EVALZ TDGL014 ISL97682IRTZEVALZ LM3508TLEV EA6358NH MAX16826EVKIT MAX16839EVKIT+ TPS92315EVM-516 MAX1698EVKIT MAX6956EVKIT+ OM13321,598 DC986A DC909A DC824A STEVAL-LLL006V1 IS31LT3948-GRLS4-EB 104PW03F PIM526 PIM527 MAX6946EVKIT+ MAX20070EVKIT# MAX21610EVKIT# MAX20090BEVKIT# MAX20092EVSYS# PIM498 AP8800EV1 ZXLD1370/1EV4 MAX6964EVKIT MAX25240EVKIT# MAX25500TEVKITC# MAX77961BEVKIT06# 1216.1013 TPS61176EVM-566 TPS61197EVM TPS92001EVM-628 1270 1271.2004 1272.1030 1273.1010 1278.1010 1279.1002 1279.1001 1282.1000 1293.1900 1293.1800 1293.1700 1293.1500