austriamicrosystems AG

is now

ams AG

The technical content of this austriamicrosystems datasheet is still valid.

Contact information:

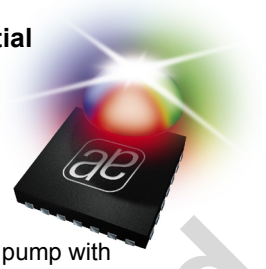
Headquarters:
ams AG
Tobelbaderstrasse 30
8141 Unterpremstaetten, Austria
Tel: +43 (0) 3136 500 0
e-Mail: ams_sales@ams.com

Please visit our website at **www.ams.com**

AS3665

Datasheet, Confidential

9 Channel Advanced Command Driven RGB/White LED Driver



1 General Description

The AS3665 is a capacitive low noise charge pump with 9 current sources. The charge pump automatically switches between 1:1 and 1:1.5 modes. The connected current sources have a very low voltage compliance to improve efficiency of the whole system. Three current sources have the possibility to operate either from VBAT or VCP (especially useful for red LEDs).

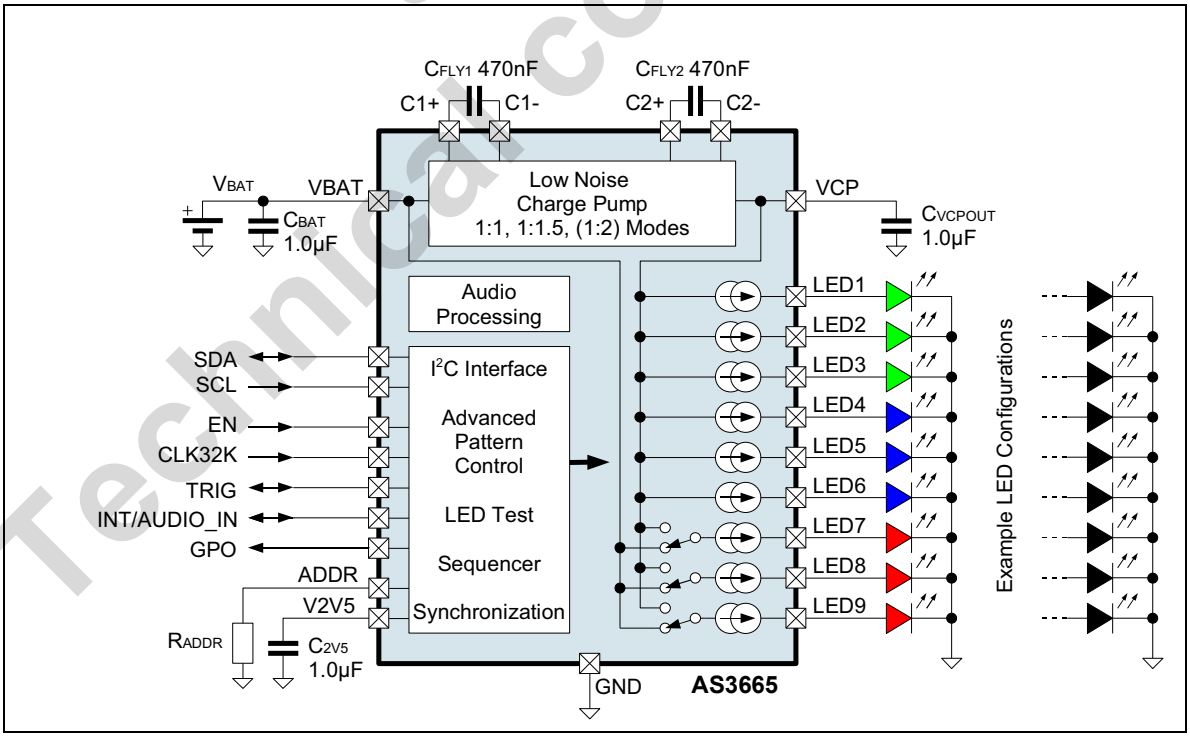
The internal control is done by command based pattern generators implemented by three sequencers. These commands are optimized for lighting applications (e.g. ramp up brightness logarithmically). It includes high level commands like conditionals jumps and variables. Any of the three sequencers can be dynamically mapped to any of the 9 PWM generators for the LEDs.

The AS3665 supports an audio input and sophisticated light patterns can be controlled by internal digital filters.

The AS3665 is controlled by I^2C mode. Synchronization over several AS3665 is possible by the TRIG pin.

The AS3665 is available in a space-saving WL-CSP-25 (2.610x2.675mm) 0.5mm pitch and operates over the -30ºC to +85ºC temperature range.

Figure 1. Typical Operating Circuit



2 Key Features

- High efficiency capacitive 150mA charge pump with 1:1, 1:1.5 and 1:2 modes with automatic mode switching; 1:2 mode can be disabled
- 9 Channel High Side 20mA Current sources
  - Less than 50mV at 10mA dropout voltage
  - LED7,8,9 either powered by VBAT or VCP
- Advanced Command based Pattern Generator
  - 96 x 16 bits program memory
  - Dedicated lighting commands like logarithmic fade
  - Programming control and conditional jumps
- Audio Controlled Lighting with internal digital filters
- 3 Sequencers
  - Dynamically mapped to 9 PWM generators
  - Internal/External Synchronization
- 9 PWM generators (12 bit resolution)
  - Automatic RGB Color Correction by TAMB
- I^2C interface with dedicated EN pin
- Available in WL-CSP-25 (2.610x2.675mm) 0.5mm pitch

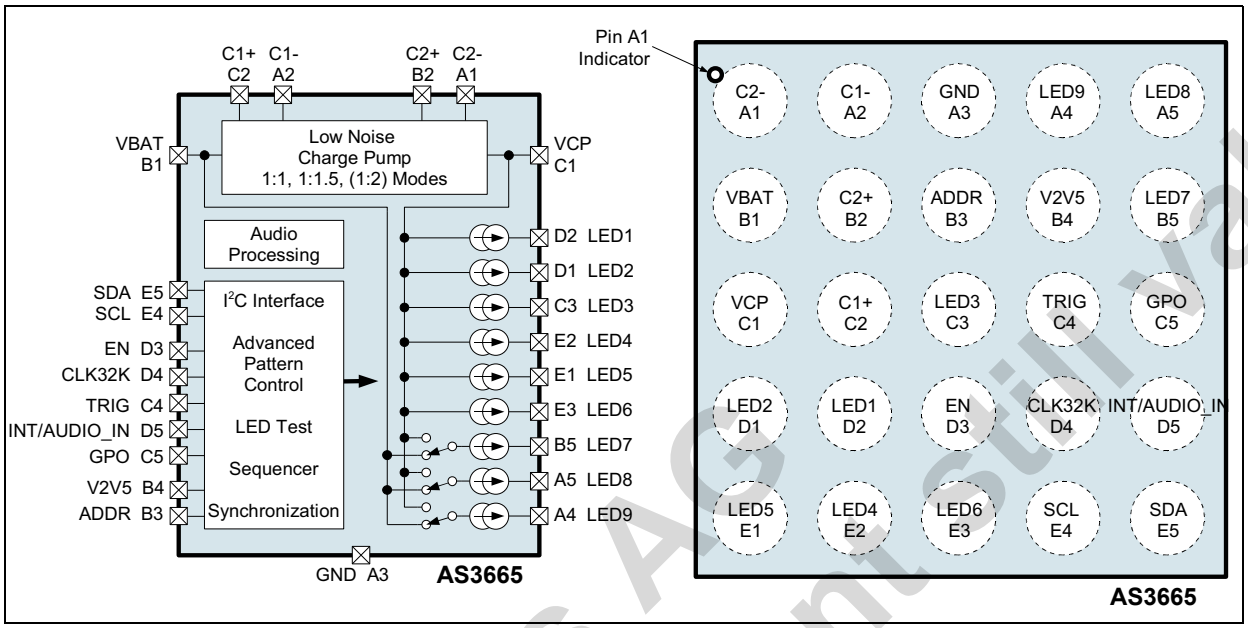
3 Applications

RGB/White Fun or Event LED for mobile phones or portable devices; Lighting Management Unit

4 Pinout

Pin Assignment

Figure 2. Pin Assignments WL-CSP-25 (2.610x2.675mm) 0.5mm pitch (Top View)



Pin Description

Table 1. Pin Description for AS3665

Pin Number	Pin Name	Description
A1	C2-	Charge Pump flying capacitor 2 - make a short connection to capacitor C_{FLY2}
A2	C1-	Charge Pump flying capacitor 1 - make a short connection to capacitor C_{FLY1}
A3	GND	Ground supply input pin
A4	LED9	LED9 output - current source from VCP or VBAT
A5	LED8	LED8 output - current source from VCP or VBAT
B1	VBAT	Positive supply input pin
B2	C2+	Charge Pump flying capacitor 2 - make a short connection to capacitor C_{FLY2}
B3	ADDR	Digital input - I^2C address select; the value of the resistor R_{ADDR} defines the actual I^2C address used
B4	C_{2V5}	Internal supply - connect a 1µF ceramic capacitor between C_{2V5} and GND
B5	LED7	LED7 output - current source from VCP or VBAT
C1	VCP	Charge Pump output - make a short connection to capacitor C_{VCPOUT}
C2	C1+	Charge Pump flying capacitor 1 - make a short connection to capacitor C_{FLY1}
C3	LED3	LED3 output - current source from VCP
C4	TRIG	Digital open drain input/output - used to synchronize across several AS3665
C5	GPO	Digital open drain input/output - General purpose output and ADC input
D1	LED2	LED2 output - current source from VCP
D2	LED1	LED1 output - current source from VCP

Table 1. Pin Description for AS3665 (Continued)

Pin Number	Pin Name	Description
D3	EN	Digital input - active high enable for AS3665
D4	CLK32K	Digital clock input - connect a 32.768kHz signal; if this signal is not available, connect this pin to GND
D5	INT/AUDIO_IN	Depending on the AS3665 configuration INT/AUDIO_IN is a 1. Open drain digital output - interrupt output pin 2. Analog input - audio or ADC signal input
E1	LED5	LED5 output - current source from VCP
E2	LED4	LED4 output - current source from VCP
E3	LED6	LED6 output - current source from VCP
E4	SCL	Digital input - clock input for I^2C communication
E5	SDA	Digital open drain input/output - data input/output for I^2C communication

5 Absolute Maximum Ratings

Stresses beyond those listed in Table 2 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 Table 3, “Electrical Characteristics,” on page 5 is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Table 2. 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	**Note:** Diode between VCP and VBAT
LED1, LED2...LED9 to GND	-0.3	VCP + 0.3 7.0	V	
SDA, SCL, EN, CLK32K, TRIG, INT/AUDIO_IN, GPO, ADDR, C_{2V5} to GND	-0.3	VBAT + 0.3 7.0	V	
Input Pin Current without causing latchup	-100	+100 +I_{IN}	mA	Norm: EIA/JESD78
Continuous Power Dissipation (T_A = +70ºC)				
Continuous power dissipation		0.78	mW	P_T[1]
Continuous power dissipation derating factor		14.2	mW/ºC	P_{DERATE}[2]
Electrostatic Discharge				
ESD HBM		±1000	V	Norm: JEDEC JESD22-A114F
ESD CDM		±500	V	Norm: JEDEC JESD 22-C101C
ESD MM		±200	V	Norm: JEDEC JESD 22-A115-A level A
Temperature Ranges and Storage Conditions				
Junction Temperature		+150	ºC	Internally limited (overtemperature protection)
Storage Temperature Range	-55	+125	ºC	
Humidity	5	85	%	Non condensing
Body Temperature during Soldering		+260	ºC	according to IPC/JEDEC J-STD-020C

1. Depending on actual PCB layout and PCB used
2. P_{DERATE} derating factor changes the total continuous power dissipation (P_T) if the ambient temperature is not 70ºC. Therefore for e.g. T_{AMB}=85ºC calculate P_T at 85ºC = P_T - P_{DERATE} * (85ºC - 70ºC)

6 Electrical Characteristics

V_{VBAT} = +2.7V to +5.5V, T_{AMB} = -30ºC to +85ºC, unless otherwise specified. Typical values are at V_{VBAT} = +3.6V, T_{AMB} = +25ºC, unless otherwise specified.

Table 3. Electrical Characteristics

Symbol	Parameter	Condition		Min	Typ	Max	Unit
General Operating Conditions							
V_{VBAT}	Supply Voltage			2.7	3.6	5.5	V
$V_{VBATREDUCED\_FUNC}$	Supply Voltage	AS3665 functionally working, but not all parameters fulfilled		2.5		2.7	V
$I_{SHUTDOWN}$	Shutdown Current				0.4	1.3	µA
I_{STANBY}	Standby mode Current	I^2C interface active			1.6	6.0	µA
I_{ACTIVE}	Active mode Current	I^2C interface active Internal oscillator running, program executed			300		µA
$I_{CP1:1.5}$	Charge Pump Current	Charge pump operating in 1:1.5 mode, no load current			0.7		mA
T_{AMB}	Operating Temperature			-30	25	85	ºC
Charge Pump							
V_{VOUT}	Charge Pump output Voltage (pin VOUT)	Internally Limited				5.5	V
I_{VOUT}	Charge Pump output current			0.0		150	mA
η	Efficiency				75		%
f_{CLK}	Operating Frequency	All internal timings are derived from this oscillator if no clock is applied on pin CLK32K		-10%	2.0	+10%	MHz
R_{CP}	Charge pump effective resistance	V_{VBAT}>=3.3V, I_{LED}=100mA	1:1 Mode		0.65		Ω
			1:1.5 Mode		3.3		Ω
Current Sources							
$I_{LED1..9}$	LED1...LED9 output current range			0.0		25.5	mA
$I_{LED1..9\Delta}$	LED1...LED9 current source accuracy	I_{LED} = 17.5mA		-7		+7	%
$I_{LED1..9\ MATCH}$	LED1...LED9 current source matching	I_{LED} = 17.5mA			2.5		%
$I_{LED1..9\ LEAKAGE}$	LED1...LED9 leakage current	current source off		-5	0	+5	µA
V_{ILED_COMP}	LED1...LED9 current source voltage compliance	Minimum voltage between pin VOUT and LED1...LED9 or VBAT and LED7...LED9				100	mV
ADC							
ADC_{RES}	ADC resolution				10		Bits
ADC_{INL}	ADC Integral non-linearity			-2	±0.2	+2	LSB
ADC_{DNL}	ADC differential non-linearity			-2	±0.25	+2	LSB
ADC_{LSB}	LSB of ADC conversion				6.1		mV

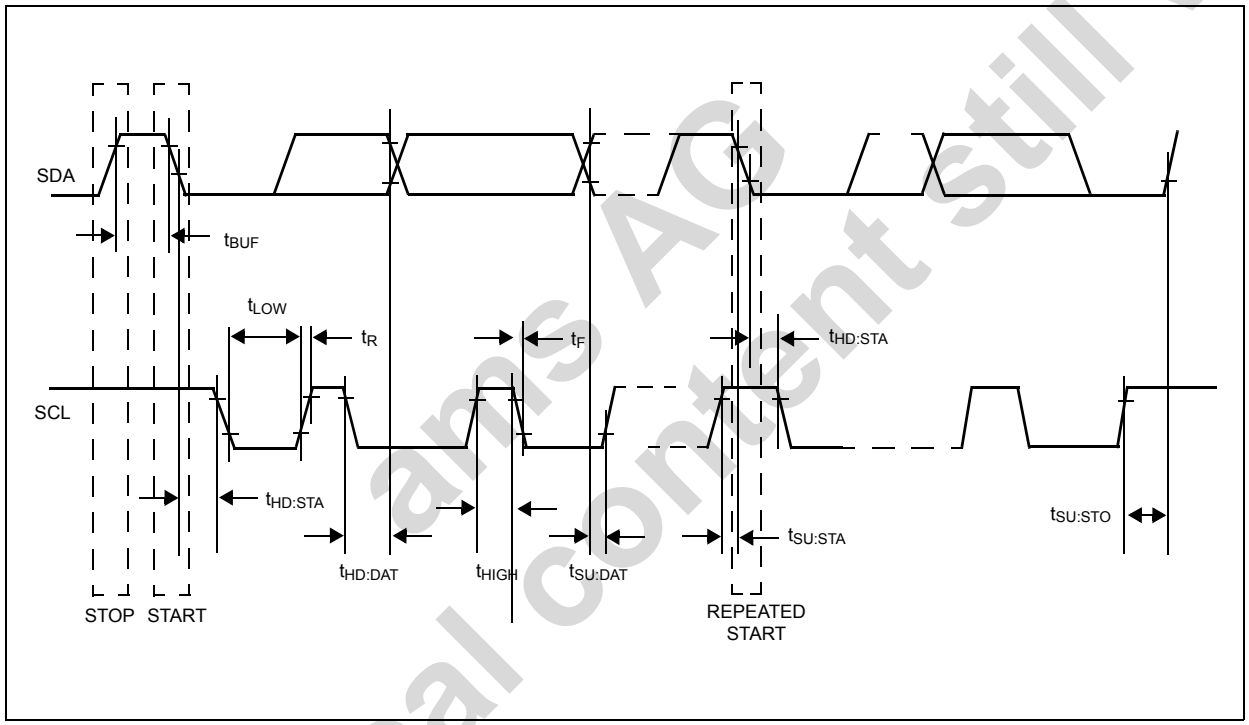
Table 3. Electrical Characteristics (Continued)

Symbol	Parameter	Condition	Min	Typ	Max	Unit
$ADC_{TOFFSET}$	ADC temperature measurement offset value			393		°C
ADC_{TC}	Code temperature coefficient			1.322		°C/ Code
T_{TOL}	Temperature sensor accuracy		-10		+10	°C
Audio Input						
R_{AUDIO_IN}	Audio Input resistance	pin INT/AUDIO_IN if used as analog input; at maximum input gain (+30dB)		20		kΩ
Digital Interface						
V_{IH}	High Level Input Voltage	Pins SDA, SCL, EN, CLK32K, TRIG, INT/AUDIO_IN, GPO[1]	1.26		V_{VBAT}	V
V_{IL}	Low Level Input Voltage		0.0		0.54	V
V_{OL}	Low Level Output Voltage	Pins SDA, TRIG, INT/AUDIO_IN, GPO I_{OL}=3mA			0.2	V
I_{LEAK}	Leakage Current	Pins SDA, SCL, EN, CLK32K, TRIG, INT/ AUDIO_IN, GPO		0.01	1.0	µA
I^2C mode timings - see Figure 3 on page 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]		0		0.9	µs
$t_{SU:DAT}$	Data Setup Time[4]		100			ns
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		20 + $0.1C_B$		300	ns
$t_{SU:STO}$	Setup Time for STOP Condition		0.6			µs
C_B	Capacitive Load for Each Bus Line	C_B — total capacitance of one bus line in pF			400	pF
$C_{I/O}$	I/O Capacitance (SDA, SCL)				10	pF
$t_{TIMEOUT}$	I^2C timeout	If SCL and SDA are low for longer than this time, the AS3665 is switched into shutdown mode[5]		100		ms

1. The logic input levels V_{IH} and V_{IL} allow for 1.8V supplied driving circuit
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.
5. This feature can be disabled by setting auto_shutdown (see page 13)=0

Timing Diagrams

Figure 3. I^2C mode Timing Diagram



7 Typical Operating Characteristics

V_{VBAT} = 3.6V, T_A = +25ºC (unless otherwise specified).

Figure 4. Efficiency vs. Battery voltage, I_{LEDS}=50mA

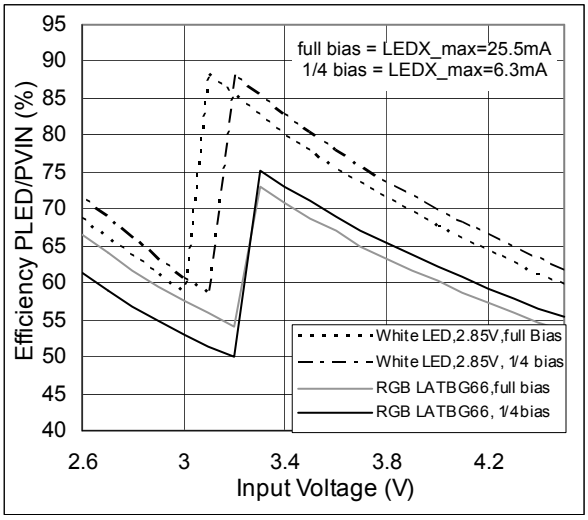


Figure 5. I_{VBAT} vs. Battery voltage, I_{LEDS}=50mA

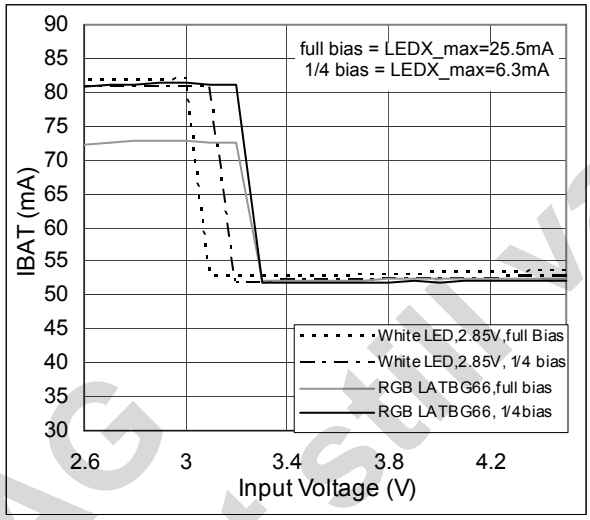


Figure 6. I_{LEDS} vs. Battery voltage

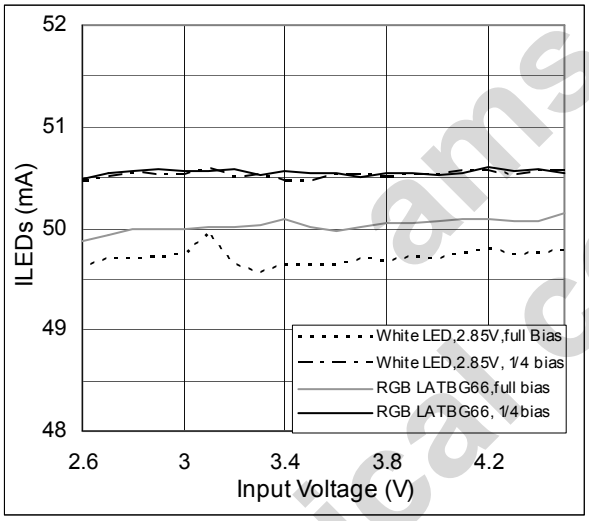


Figure 7. I_{LED1} Linearity of current source vs. Code

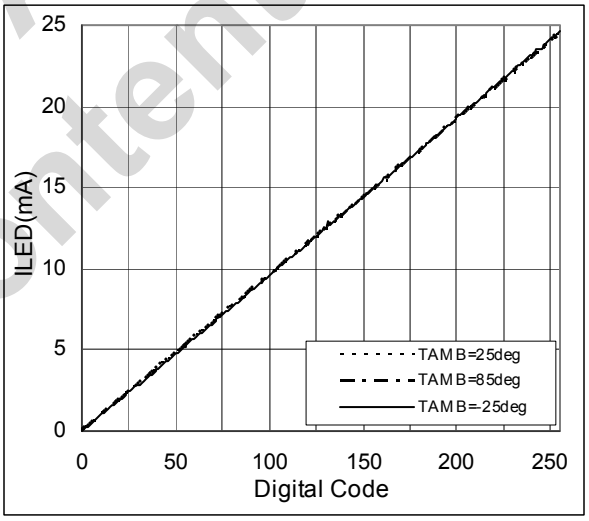


Figure 8. I_{LED1} Monotony of current source vs. Code

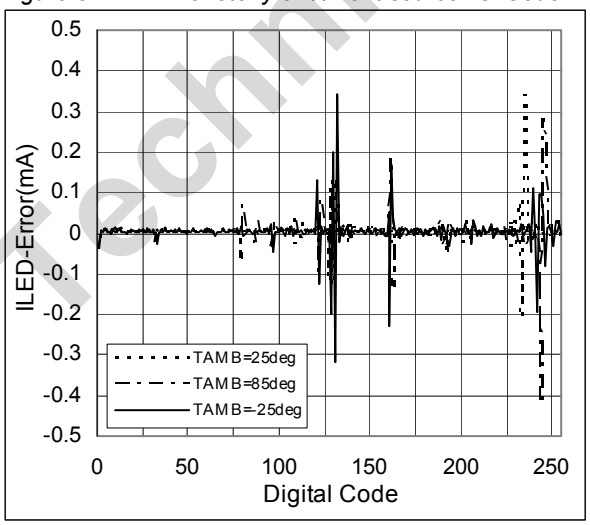


Figure 9. Logarithmic PWM ramp

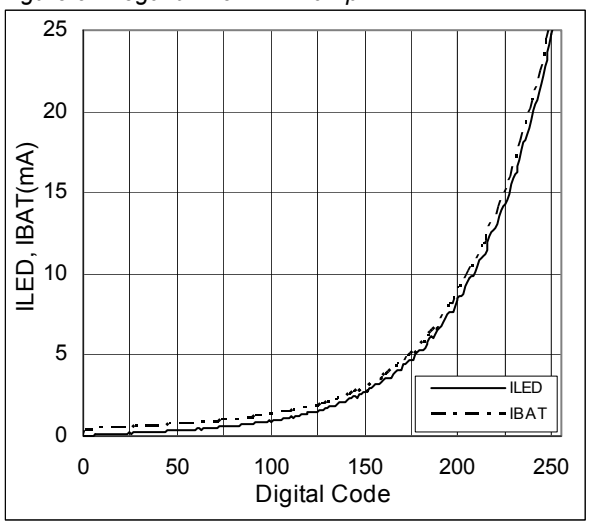


Figure 10. Logarithmic PWM ramp

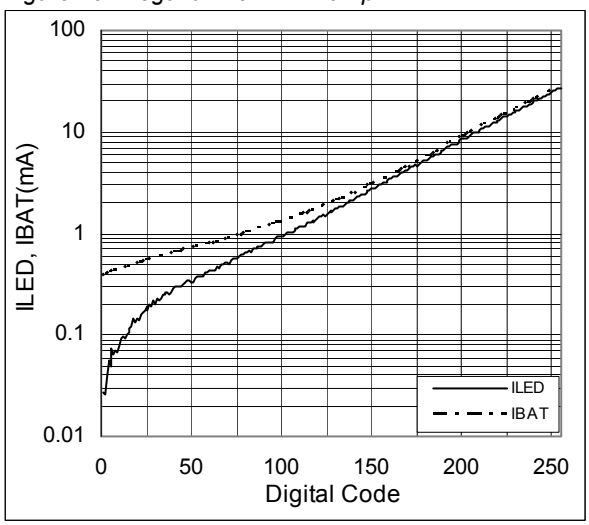


Figure 11. ILED vs. Voltage on current source

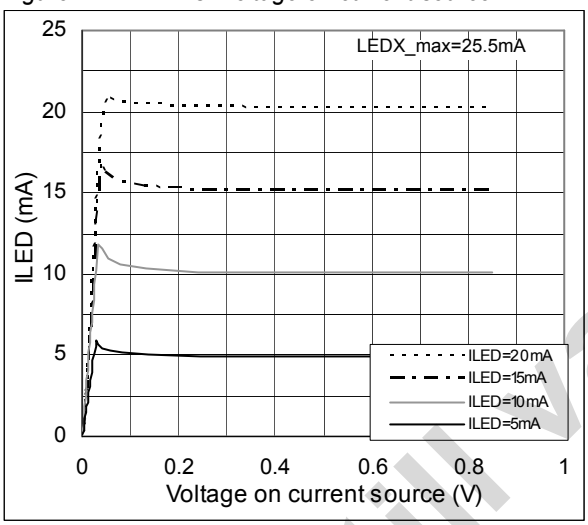


Figure 12. ILED vs. Voltage on current source

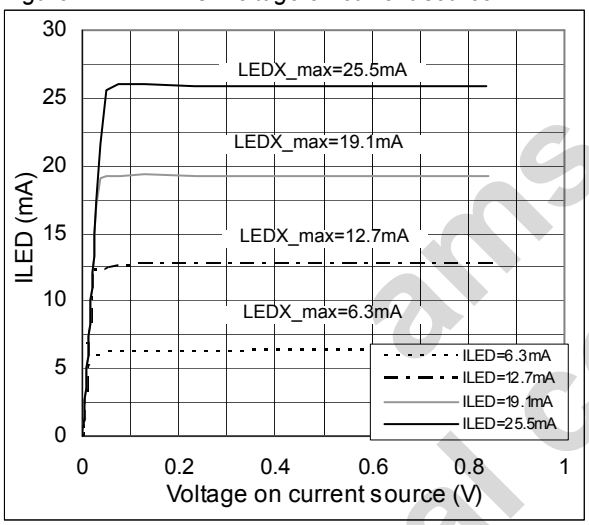


Figure 13. ILED vs. Voltage on current source

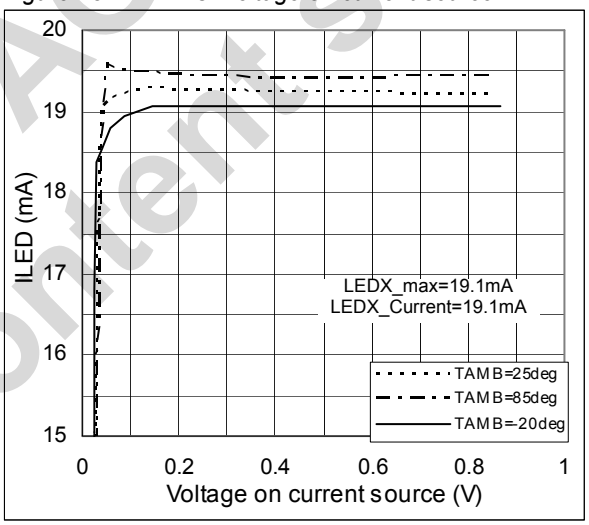
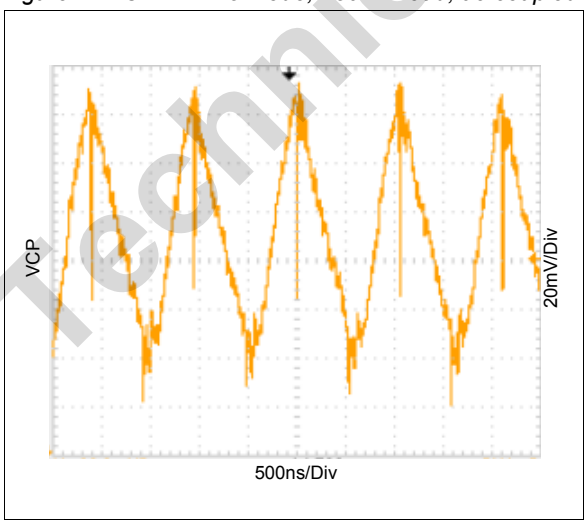


Figure 14. CP in 1:1.5 mode, 150mA load, ac-coupled



8 Detailed Description

The AS3665 is a fixed frequency charge pump. Its output (VOUT) is connected to nine current sources (LED1..LED9). A sophisticated command based pattern generator with three sequencers controls the nine PWM generators (12 bit resolution), which are connected to the current sources.

Commands are downloaded to the AS3665 internal memory space and can be executed autonomously in the three sequencers. The commands are optimized for lighting applications (e.g. a single command executes logarithmic up dimming). It supports command flow control (like unconditional and conditional jumps). Variables which are accessible through the I^2C interface allow control of the program execution by the I^2C interface and communication between the three sequencers.

The three sequencers can be dynamically assigned to any of the nine outputs (under program control).

The AS3665 supports an audio input pin INT/AUDIO_IN which allows the control of patterns depending on an audio input signal. This audio input can be feed through internal digital filters for better visual appearance.

If the audio feature is not used, the pin INT/AUDIO_IN can be used as interrupt output[1] to send interrupts.

The AS3665 is controlled by an I^2C interface and additional dedicated control lines. An EN input operates as a global enable/disable pin and with the pin TRIG several AS3665 can be synchronized in a system. A separate CLK32K input can be used to set an exact clock input frequency (all internal timings can be derived either from CLK32K or an internal oscillator). The I^2C address is selectable by the pin ADDR - see I^2C Address selection on page 40. A GPO pin can be used for external control or as an additional ADC input.

The AS3665 supports LED testing (verification of the performance of the connected LEDs in an assembled system).

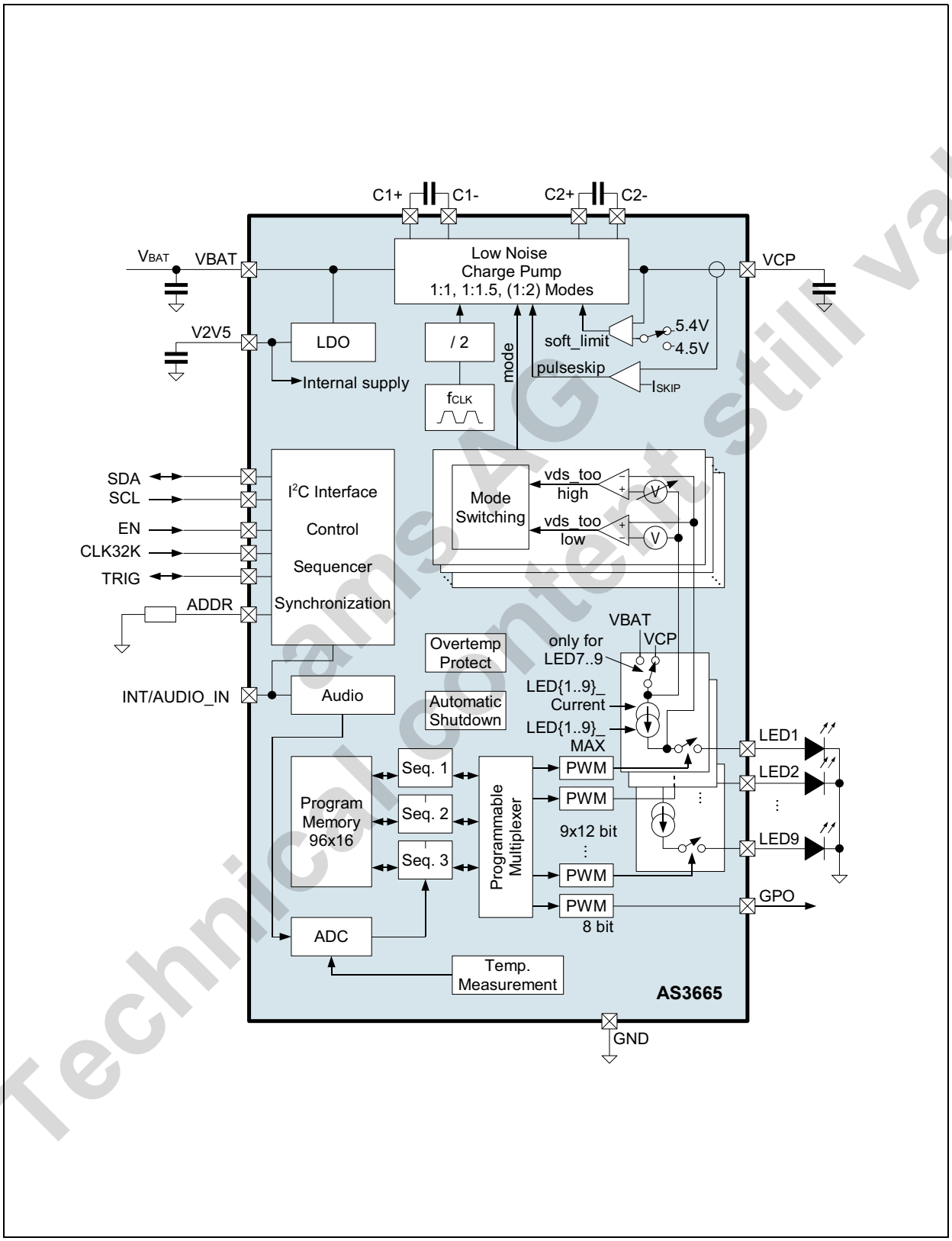
Following blocks are included inside the AS3665:

- Low Noise charge pump operating in 1:1, 1:1.5 and 1:2
- Automatic mode switching of the charge pump (up & down)
- 1MHz oscillator
- Internal LDO for powering the internal circuitry
- Audio processing of an analog input signal
- Overtemperature Protection
- Temperature Measurements of the AS3665
- 10 Bit ADC
- 9x12 bit, 1x8 bit PWM Generators
- 6 accurate current sources connected to VCP
- 3 accurate current source configurable to be connected to VBAT or VCP (to improve efficiency e.g. of red LEDs)
- Internal memory for the program execution
- 3 sequencers (3 parallel processing units)
- a fully programmable multiplexer connecting the three sequencers to the 10 PWM generators
- Automatic shutdown to safe power (if SCL and SDA=0 for 100ms)

1. INT/AUDIO_IN is an open drain output. Several interrupt can be easily combined externally.

Internal Circuit

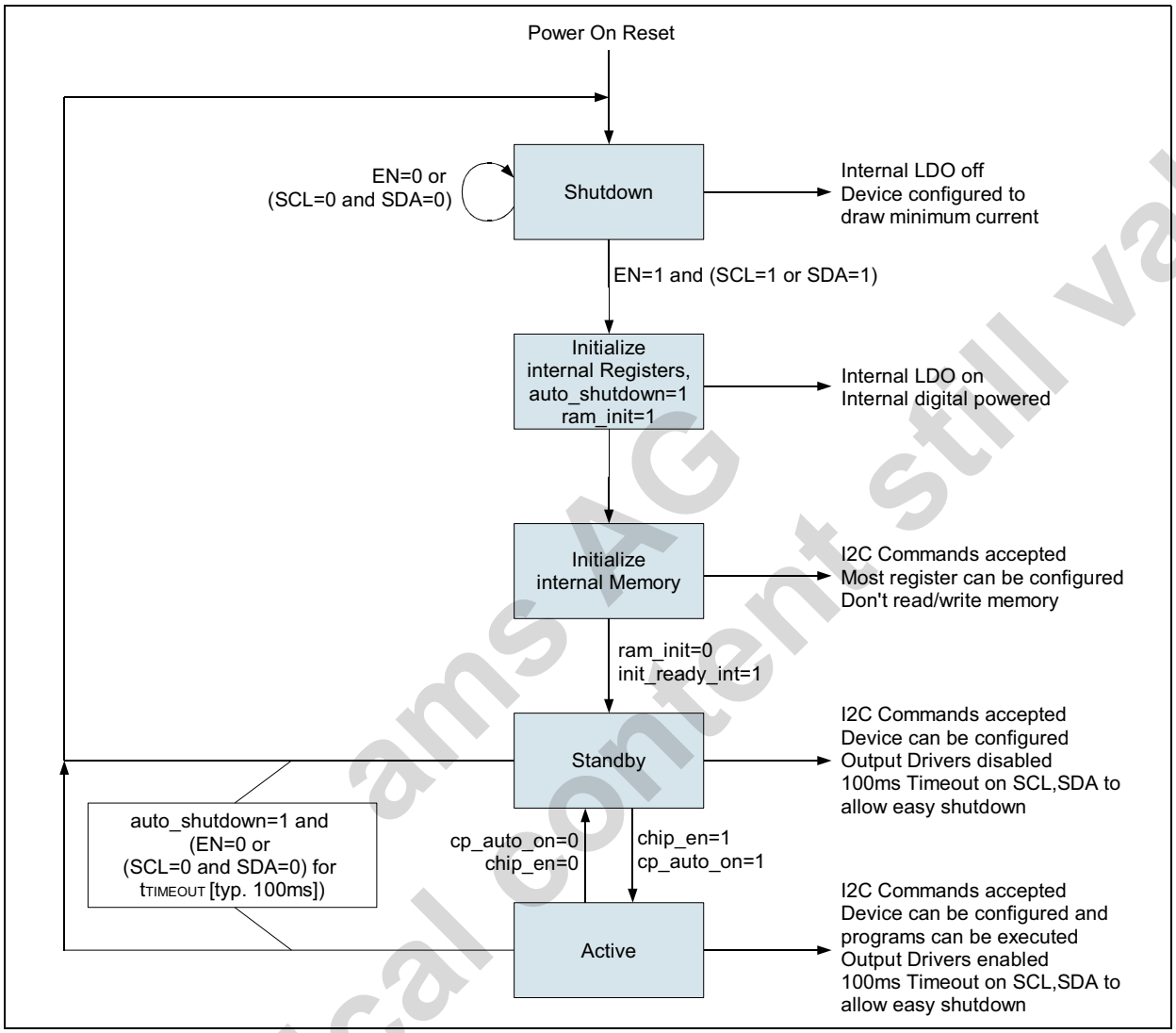
Figure 15. AS3665 internal circuit



Device Operating Mode

The operating mode is selected according to the following flowchart:

Figure 16. AS3665 operating mode selection



After power on reset, the AS3665 waits until EN=1 and SCL=1 or SDA=1[2] and then initializes its internal registers and program memory. Once standby mode is reached, the program and setup can be download to the AS3665 and by setting chip_en=1 the program can be executed.

2. SCL and SDA is monitored to detect if the I^2C bus is powered. Therefore if EN is not used, it can be tied to VBAT and the mode selection between shutdown and the other modes is performed by SCL and SDA.

If EN is pulled low or if the power from the I²C bus pullup resistors is removed[3] for more than $t_{TIMEOUT}$, the AS3665 enters shutdown[4].

Table 4. Exec_Enable Register

Addr: 00h		Exec_Enable Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
6	chip_en	0h	R/W	Enables the active mode (see Figure 16)	
				0	AS3665 standby mode select. Set cp_auto_on=0 before setting chip_en=0. Output drivers disabled, I²C communication possible
				1	AS3665 active mode select. Set cp_auto_on=1 after setting chip_en=1 All functions active, internal oscillator running.
7	ram_init	0h	R/W	Initialization of the internal memory (see Figure 16)	
				0	Memory initialization is finished
				1	Writing: Reset internal program memory and all register from 60h...FFh to their default state Reading: memory initialization ongoing; when finished an interrupt can be triggered (init_ready_int (see page 37) is set)

The bit auto_shutdown controls the automatic entering of shutdown mode if the I²C bus is disabled:

Table 5. Supervision Register

Addr: 08h		Supervision Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
7	auto_shutdown	1h	R/W	Enables the shutdown mode (see Figure 16)	
				0	AS3665 cannot enter shutdown do not set pin EN=0 if cp_auto_on=1 or cp_on=1
				1	AS3665 can use shutdown EN=0 can be used to enter shutdown mode

A complete reset cycle can be triggered by setting bit force_reset:

Table 6. Reset_Control Register

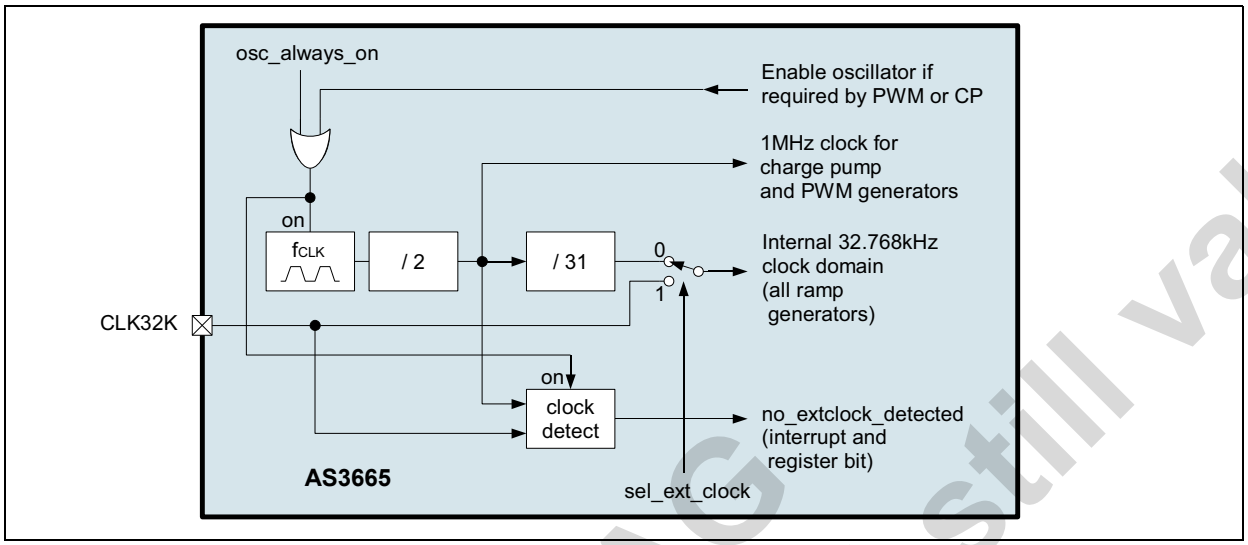
Addr: 3Ch		Reset_Control Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
0	force_reset	0	R/W	Start reset cycle (see Figure 16)	
				0	Normal operation
				1	Reset all registers from 00h...1Fh and 5Fh to their default value

3. Therefore SCL and SDA both are low.
4. Unless auto_shutdown (see page 13)=0

Clock Generation

The AS3665 has an internal oscillator running at fCLK and an external clock input CLK32K:

Figure 17. Clock Generation



The charge pump and the PWM generator use the fCLK clock signal from the internal oscillator. Depending on the signal sel_ext_clock, the internal timers and ramp generators use either the pin CLK32K as input or fCLK divided by 2 and 31:

Table 7. GPO_Control Register

Addr: 04h		GPO_Control Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
6	sel_ext_clock	0h	R/W	Enables the external clock on CLK32K (see Figure 17)	
				0	Use internal fCLK clock divided by 31*2
				1	Use external clock on CLK32K (also osc_always_on=0)[1]

1. Using an external clock has two advantages:
   a) Reduced quiescent current: the internal clock is switched off whenever possible and the timers run from CLK32K.
   b) All timings (e.g. ramp-up, wait) are as accurate as the external clock (usually derived from a crystal).

The external clock on CLK32K is monitored and if the internal clock is enabled and no valid clock are detected the register bit no_extclock_detected (see page 37) is set and an interrupt can be triggered.

The internal oscillator is enabled and disabled automatically if register bit osc_always_on is reset:

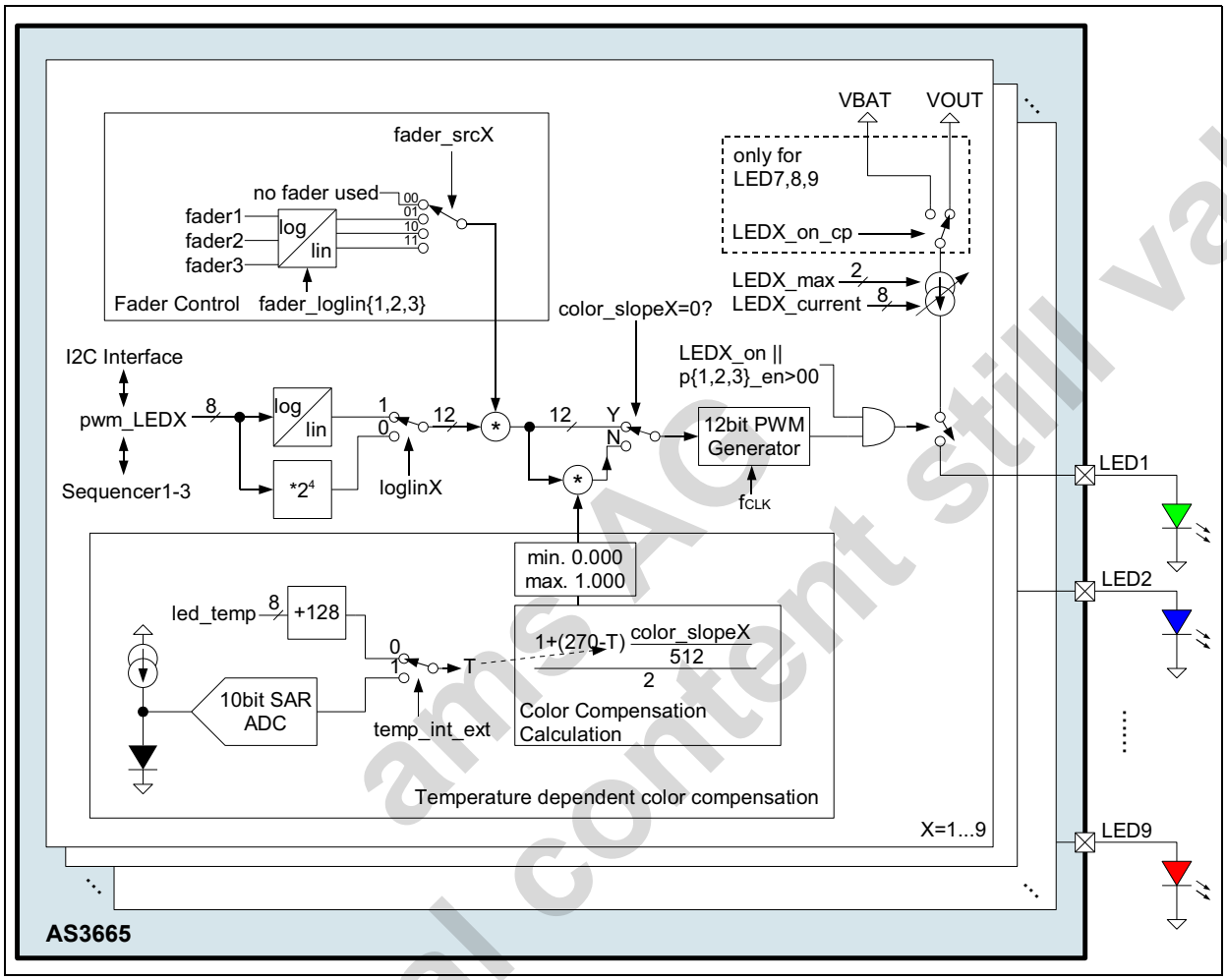
Table 8. Supervision Register

Addr: 08h		Supervision Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
5	osc_always_on	0h	R/W	Enables the internal oscillator (see Figure 17)	
				0	Enable internal oscillator only if required
				1	The internal oscillator is always running (except in shutdown mode)

Current Sources

The internal circuit of the current sources is shown in Figure 18 (one current source shown; internally there are 9 identical blocks):

Figure 18. Current Sources



The processing path consists of the following step (using current source 1 as example):

1. The input of the complete current source block is the register pwm_LED1 (see page 22). This register can be controlled by I²C directly or by any of the three sequencers (see section Sequencers on page 48).
2. The signal is converted from logarithmic domain to linear domain (depending on signal loglin1 (see page 25)) or multiplied by 16 to obtain 12 bits.
3. It passes an adjustable fader (it can be multiplied by any of the fader registers fader1, fader2 or fader3). If fader_src1 (see page 25)=0, the fader is not used (signal is unchanged).
4. Color correction is performed (temp_int_ext (see page 24) selects either internal temperature measurement or use the register led_temp (see page 24)). The gain of the color correction can be adjusted by color_slope1 (see page 25). If color_slope1=0, color correction is disabled.
5. The resulting 12 bit signal goes to the PMW generator and then to the current source itself.
6. The current source is enabled by LED1_on[5] and its current is adjusted by LED_current1 and LED1_max.

5. LED1_on...LED9_on have only effect if all sequencer are switched off (p1_en (see page 46)=00 and p2_en=00 and p3_en=00). This allow direct control of the LEDs if no program is executed.

7. LED7, LED8 and LED9 have the option to be powered by VBAT directly (configured by LED7_on_cp...LED9_on_cp)

Interface to sequencers

pwm_LED1 (see page 22), pwm_LED2...pwm_LED9 is the input PWM value of the current sources (8 bit value). This value can be either controlled by the I^2C interface or by any of the sequencers (see section Sequencers on page 48).

Logarithmic/Linear Ramping

All current sources support logarithmic or linear ramping (selected by register bits loglin1 (see page 25), loglin2...loglin9). As light is perceived logarithmically, it is recommended to keep the current sources in logarithmic mode (default setting).

RGB Color Correction

The RGB Color correction changes the output PWM value depending on the temperature (either the junction temperature if temp_int_ext (see page 24)=0, or a I^2C value stored in led_temp (see page 24) if temp_int_ext=1). This compensates different temperature drifts of LEDs and keep the white point over temperature. The slope of this temperature compensation is adjustable with the register color_slope1 (see page 25), color_slope2...color_slope9 (set to 0 if the color correction is not used).

Faders

There are three global faders: fader1 (see page 23), fader2 and fader3. Each current source can be configured to be multiplied by any of the three faders (controlled by fader_src1 (see page 25), fader_src2...fader_src9). Therefore a fader can operate on any number of current sources in parallel (e.g. to generate smooth fade-out effects on several LEDs). The faders can operate linear or logarithmic (defined by fader_loglin1 (see page 23), fader_loglin2 and fader_loglin3).

Analog Current Setting

All current sources can be completely enabled/disable by the register LED1_on, LED2_on...LED9_on. The actual analog current is set by LED_current1 (see page 17), LED_current2...LED_current9. The maximum current driving capability of the current sources is set by registers LED1_max (see page 20), LED2_max...LED9_max[6].

Current Source Registers

Analog Current setting registers

Table 9. LED_Control1 Register

Addr: 02h		LED_Control1 Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
0	LED1_on	0b	R/W	0	LED1 is off
				1	LED1 is enabled
1	LED2_on	0b	R/W	0	LED2 is off
				1	LED2 is enabled
2	LED3_on	0b	R/W	0	LED3 is off
				1	LED3 is enabled
3	LED4_on	0b	R/W	0	LED4 is off
				1	LED4 is enabled
4	LED5_on	0b	R/W	0	LED5 is off
				1	LED5 is enabled

6. Always use the minimum setting for LED1_max, LED2_max...LED9_max suitable for the application to reduce quiescent current of the internal current source

Table 9. LED_Control1 Register

Addr: 02h		LED_Control1 Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
5	LED6_on	0b	R/W	0	LED6 is off
				1	LED6 is enabled
6	LED7_on	0b	R/W	0	LED7 is off
				1	LED7 is enabled
7	LED8_on	0b	R/W	0	LED8 is off
				1	LED8 is enabled

Table 10. LED_Control2 Register

Addr: 03h		LED_Control2 Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
0	LED9_on	0b	R/W	0	LED9 is off
				1	LED9 is enabled

Table 11. LED_Current1 Register

Addr: 10h		LED_Current1 Register						
Bit	**Bit Name**	**Default**	**Access**	**Description**				
7:0	LED_current1	00h	R/W	Sets the current for current source on LED1				
					LED1_max			
					00	01	10	11
				0	Current source off			
				1	0.1mA	74.9µA	49.8µA	24.7µA
				...				
				255	25.5mA	19.1mA	12.7mA	6.3mA

Table 12. LED_Current2 Register

Addr: 11h		LED_Current2 Register						
Bit	**Bit Name**	**Default**	**Access**	**Description**				
7:0	LED_current2	00h	R/W	Sets the current for current source on LED2				
					LED2_max			
					00	01	10	11
				0	Current source off			
				1	0.1mA	74.9µA	49.8µA	24.7µA
				...				
				255	25.5mA	19.1mA	12.7mA	6.3mA

Table 13. LED_Current3 Register

Addr: 12h		LED_Current3 Register						
Bit	**Bit Name**	**Default**	**Access**	**Description**				
7:0	LED_current3	00h	R/W	Sets the current for current source on LED3				
					LED3_max			
					00	01	10	11
				0	Current source off			
				1	0.1mA	74.9µA	49.8µA	24.7µA
				...				
				255	25.5mA	19.1mA	12.7mA	6.3mA

Table 14. LED_Current4 Register

Addr: 13h		LED_Current4 Register						
Bit	**Bit Name**	**Default**	**Access**	**Description**				
7:0	LED_current4	00h	R/W	Sets the current for current source on LED4				
					LED4_max			
					00	01	10	11
				0	Current source off			
				1	0.1mA	74.9µA	49.8µA	24.7µA
				...				
				255	25.5mA	19.1mA	12.7mA	6.3mA

Table 15. LED_Current5 Register

Addr: 14h		LED_Current5 Register						
Bit	**Bit Name**	**Default**	**Access**	**Description**				
7:0	LED_current5	00h	R/W	Sets the current for current source on LED5				
					LED5_max			
					00	01	10	11
				0	Current source off			
				1	0.1mA	74.9µA	49.8µA	24.7µA
				...				
				255	25.5mA	19.1mA	12.7mA	6.3mA

Table 16. LED_Current6 Register

Addr: 15h		LED_Current6 Register						
Bit	**Bit Name**	**Default**	**Access**	**Description**				
7:0	LED_current6	00h	R/W	Sets the current for current source on LED6				
					LED6_max			
					00	01	10	11
				0	Current source off			
				1	0.1mA	74.9µA	49.8µA	24.7µA
				...				
				255	25.5mA	19.1mA	12.7mA	6.3mA

Table 17. LED_Current7 Register

Addr: 16h		LED_Current7 Register						
Bit	**Bit Name**	**Default**	**Access**	**Description**				
7:0	LED_current7	00h	R/W	Sets the current for current source on LED7				
					LED7_max			
					00	01	10	11
				0	Current source off			
				1	0.1mA	74.9µA	49.8µA	24.7µA
				...				
				255	25.5mA	19.1mA	12.7mA	6.3mA

Table 18. LED_Current8 Register

Addr: 17h		LED_Current8 Register						
Bit	**Bit Name**	**Default**	**Access**	**Description**				
7:0	LED_current8	00h	R/W	Sets the current for current source on LED8				
					LED8_max			
					00	01	10	11
				0	Current source off			
				1	0.1mA	74.9µA	49.8µA	24.7µA
				...				
				255	25.5mA	19.1mA	12.7mA	6.3mA

Table 19. LED_Current9 Register

Addr: 18h		LED_Current9 Register						
Bit	**Bit Name**	**Default**	**Access**	**Description**				
7:0	LED_current9	00h	R/W	Sets the current for current source on LED9				
					LED9_max			
					00	01	10	11
				0	Current source off			
				1	0.1mA	74.9µA	49.8µA	24.7µA
				...				
				255	25.5mA	19.1mA	12.7mA	6.3mA

Table 20. LED_MaxCurr1 Register

Addr: 19h		LED_MaxCurr1 Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
1:0	LED1_max	00b	R/W	Sets the maximum current for current source on LED1 (see LED_current1 on page 17)	
				00	I_{LED1} = 0...25.5mA
				01	I_{LED1} = 0...19.1mA
				10	I_{LED1} = 0...12.7mA
				11	I_{LED1} = 0...6.3mA
3:2	LED2_max	00b	R/W	Sets the maximum current for current source on LED2 (see LED_current2 on page 17)	
				00	I_{LED2} = 0...25.5mA
				01	I_{LED2} = 0...19.1mA
				10	I_{LED2} = 0...12.7mA
				11	I_{LED2} = 0...6.3mA
5:4	LED3_max	00b	R/W	Sets the maximum current for current source on LED3 (see LED_current3 on page 18)	
				00	I_{LED3} = 0...25.5mA
				01	I_{LED3} = 0...19.1mA
				10	I_{LED3} = 0...12.7mA
				11	I_{LED3} = 0...6.3mA
7:6	LED4_max	00b	R/W	Sets the maximum current for current source on LED4 (see LED_current4 on page 18)	
				00	I_{LED4} = 0...25.5mA
				01	I_{LED4} = 0...19.1mA
				10	I_{LED4} = 0...12.7mA
				11	I_{LED4} = 0...6.3mA

Table 21. LED_MaxCurr2 Register

Addr: 1Ah		LED_MaxCurr2 Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
1:0	LED5_max	00b	R/W	Sets the maximum current for current source on LED5 (see LED_current5 on page 18)	
				00	I_{LED5} = 0...25.5mA
				01	I_{LED5} = 0...19.1mA
				10	I_{LED5} = 0...12.7mA
				11	I_{LED5} = 0...6.3mA
3:2	LED6_max	00b	R/W	Sets the maximum current for current source on LED6 (see LED_current6 on page 19)	
				00	I_{LED6} = 0...25.5mA
				01	I_{LED6} = 0...19.1mA
				10	I_{LED6} = 0...12.7mA
				11	I_{LED6} = 0...6.3mA
5:4	LED7_max	00b	R/W	Sets the maximum current for current source on LED7 (see LED_current7 on page 19)	
				00	I_{LED7} = 0...25.5mA
				01	I_{LED7} = 0...19.1mA
				10	I_{LED7} = 0...12.7mA
				11	I_{LED7} = 0...6.3mA
7:6	LED8_max	00b	R/W	Sets the maximum current for current source on LED8 (see LED_current8 on page 19)	
				00	I_{LED8} = 0...25.5mA
				01	I_{LED8} = 0...19.1mA
				10	I_{LED8} = 0...12.7mA
				11	I_{LED8} = 0...6.3mA

Table 22. LED_MaxCurr3 Register

Addr: 1Bh		LED_MaxCurr3 Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
1:0	LED9_max	00b	R/W	Sets the maximum current for current source on LED9 (see LED_current9 on page 20)	
				00	I_{LED9} = 0...25.5mA
				01	I_{LED9} = 0...19.1mA
				10	I_{LED9} = 0...12.7mA
				11	I_{LED9} = 0...6.3mA

PWM Data Input Registers

Table 23. PWM_LED1, PWM_LED2...PWM_LED9, PWM_GPO Registers

Addr: 80h-89h			PWM_LED1, PWM_LED2...PWM_LED9, PWM_GPO Register			
Addr	**Bit**	**Name**	**Default**	**Access**	**Description**	
80h	7:0	pwm_LED1	00h	R/W	PWM value for Current source on LED1	
					0	LED1 Off
					...	
					255	LED1 Full Scale
81h	7:0	pwm_LED2	00h	R/W	PWM value for Current source on LED2	
					0	LED2 Off
					...	
					255	LED2 Full Scale
82h	7:0	pwm_LED3	00h	R/W	PWM value for Current source on LED3	
					0	LED3 Off
					...	
					255	LED3 Full Scale
83h	7:0	pwm_LED4	00h	R/W	PWM value for Current source on LED4	
					0	LED4 Off
					...	
					255	LED4 Full Scale
84h	7:0	pwm_LED5	00h	R/W	PWM value for Current source on LED5	
					0	LED5 Off
					...	
					255	LED5 Full Scale
85h	7:0	pwm_LED6	00h	R/W	PWM value for Current source on LED6	
					0	LED6 Off
					...	
					255	LED6 Full Scale
86h	7:0	pwm_LED7	00h	R/W	PWM value for Current source on LED7	
					0	LED7 Off
					...	
					255	LED7 Full Scale
87h	7:0	pwm_LED8	00h	R/W	PWM value for Current source on LED8	
					0	LED8 Off
					...	
					255	LED8 Full Scale
88h	7:0	pwm_LED9	00h	R/W	PWM value for Current source on LED9	
					0	LED9 Off
					...	
					255	LED9 Full Scale

Table 23. PWM_LED1, PWM_LED2...PWM_LED9, PWM_GPO Registers (Continued)

Addr: 80h-89h			PWM_LED1, PWM_LED2...PWM_LED9, PWM_GPO Register			
Addr	**Bit**	**Name**	**Default**	**Access**	**Description**	
89h	7:0	pwm_GPO	00h	R/W	PWM value for GPO PWM generator (8 bits)	
					0	PWM GPO Off
					...	
					255	PWM GPO Full Scale

RGB Color correction, Fader and Logarithmic/Linear Registers

Table 24. LED_Control2 Register

Addr: 03h			LED_Control2 Register		
Bit	**Bit Name**	**Default**	**Access**	**Description**	
3	temp_comp_mode[1]	0	R/W	Temperature compensation operating mode	
				0	Normal Mode
				1	Positive Values of correction: Normal operation Negative values of correction: correction value divided by 2
4	fader_loglin1	0	R/W	Fader 1 linear / logarithmic control	
				0	Linear Operation
				1	Logarithmic Operation
5	fader_loglin2	0	R/W	Fader 2 linear / logarithmic control	
				0	Linear Operation
				1	Logarithmic Operation
6	fader_loglin3	0	R/W	Fader 3 linear / logarithmic control	
				0	Linear Operation
				1	Logarithmic Operation

1. Its safe to keep temp_comp_mode at default '0'

Table 25. Fader1, Fader2 and Fader3 Registers

Addr: 9B-9Dh			Fader1, Fader2 and Fader3 Register			
Addr	**Bit**	**Name**	**Default**	**Access**	**Description**	
9Bh	7:0	fader1	00h	R/W	Global Fader1 value	
					0	Off
					...	
					255	Full Scale
9Ch	7:0	fader2	00h	R/W	Global Fader2 value	
					0	Off
					...	
					255	Full Scale

Table 25. Fader1, Fader2 and Fader3 Registers (Continued)

Addr: 9B-9Dh			Fader1, Fader2 and Fader3 Register			
Addr	**Bit**	**Name**	**Default**	**Access**	**Description**	
9Dh	7:0	fader3	00h	R/W	Global Fader3 value	
					0	Off
					...	
					255	Full Scale

Table 26. Temp_Sense_ Control Register

Addr: 0Eh			Temp_Sense_ Control Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**		
0	temp_int_ext	0b	R/W	The RGB color correction uses internal/external source for temperature compensation (see RGB Color Correction on page 16)		
				0	I^2C register led_temp is used	
				1	internal junction temperature measured[1]	
1	temp_sens_on	0b	R/W	Internal temperature sensor enable		
				0	Internal temperature sensor off	
				1	Internal temperature sensor on	
2	temp_meas_busy	0b	R	Internal temperature sensor busy status signal		
				0	Internal temperature sensor off or not busy	
				1	Internal temperature sensor busy	

1. Set temp_sens_on=1

Table 27. LED_Temp Register

Addr: 1Fh			LED_Temp Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**		
7:0	led_temp	00h	R/W	Value used for RGB color correction if temp_int_ext=1 (see RGB Color Correction on page 16)		
				185	-30ºC	
				142	25ºC	
				96	+85ºC	

Table 28. Driver_Setup1 Register

Addr: A0h		Driver_Setup1 Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
4:0	color_slope1	00h	R/W	LED1 RGB Color Correction (see page 16) slope	
				00h	RGB Color Correction disabled
				01h	+0.15%/ºC
				...	...
				0Fh	+2.263%/ºC
				11h	-2.263%/ºC
				...	...
				1Fh	-0.15%/ºC
5	loglin1	1b	R/W	LED1 Logarithmic/Linear Ramping (see page 16)	
				0	linear ramping/dimming
				1	logarithmic ramping/dimming
7:6	fader_src1	00b	R/W	LED1 Faders (see page 16)	
				00	fader disabled
				01	use fader1 (see page 23)
				10	use fader2
				11	use fader3

Table 29. Driver_Setup2 Register

Addr: A1h		Driver_Setup2 Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
4:0	color_slope2	00h	R/W	LED2 RGB Color Correction (see page 16) slope	
				00h	RGB Color Correction disabled
				01h	+0.15%/ºC
				...	...
				0Fh	+2.263%/ºC
				11h	-2.263%/ºC
				...	...
				1Fh	-0.15%/ºC
5	loglin2	1b	R/W	LED2 Logarithmic/Linear Ramping (see page 16)	
				0	linear ramping/dimming
				1	logarithmic ramping/dimming
7:6	fader_src2	00b	R/W	LED2 Faders (see page 16)	
				00	fader disabled
				01	use fader1 (see page 23)
				10	use fader2
				11	use fader3

Table 30. Driver_Setup3 Register

Addr: A2h		Driver_Setup3 Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
4:0	color_slope3	00h	R/W	LED3 RGB Color Correction (see page 16) slope	
				00h	RGB Color Correction disabled
				01h	+0.15%/ºC
				...	...
				0Fh	+2.263%/ºC
				11h	-2.263%/ºC
				...	...
				1Fh	-0.15%/ºC
5	loglin3	1b	R/W	LED3 Logarithmic/Linear Ramping (see page 16)	
				0	linear ramping/dimming
				1	logarithmic ramping/dimming
7:6	fader_src3	00b	R/W	LED3 Faders (see page 16)	
				00	fader disabled
				01	use fader1 (see page 23)
				10	use fader2
				11	use fader3

Table 31. Driver_Setup4 Register

Addr: A3h		Driver_Setup4 Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
4:0	color_slope4	00h	R/W	LED4 RGB Color Correction (see page 16) slope	
				00h	RGB Color Correction disabled
				01h	+0.15%/ºC
				...	...
				0Fh	+2.263%/ºC
				11h	-2.263%/ºC
				...	...
				1Fh	-0.15%/ºC
5	loglin4	1b	R/W	LED4 Logarithmic/Linear Ramping (see page 16)	
				0	linear ramping/dimming
				1	logarithmic ramping/dimming
7:6	fader_src4	00b	R/W	LED4 Faders (see page 16)	
				00	fader disabled
				01	use fader1 (see page 23)
				10	use fader2
				11	use fader3

Table 32. Driver_Setup5 Register

Addr: A4h		Driver_Setup5 Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
4:0	color_slope5	00h	R/W	LED5 RGB Color Correction (see page 16) slope	
				00h	RGB Color Correction disabled
				01h	+0.15%/ºC
				...	...
				0Fh	+2.263%/ºC
				11h	-2.263%/ºC
				...	...
				1Fh	-0.15%/ºC
5	loglin5	1b	R/W	LED5 Logarithmic/Linear Ramping (see page 16)	
				0	linear ramping/dimming
				1	logarithmic ramping/dimming
7:6	fader_src5	00b	R/W	LED5 Faders (see page 16)	
				00	fader disabled
				01	use fader1 (see page 23)
				10	use fader2
				11	use fader3

Table 33. Driver_Setup6 Register

Addr: A5h		Driver_Setup6 Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
4:0	color_slope6	00h	R/W	LED6 RGB Color Correction (see page 16) slope	
				00h	RGB Color Correction disabled
				01h	+0.15%/ºC
				...	...
				0Fh	+2.263%/ºC
				11h	-2.263%/ºC
				...	...
				1Fh	-0.15%/ºC
5	loglin6	1b	R/W	LED6 Logarithmic/Linear Ramping (see page 16)	
				0	linear ramping/dimming
				1	logarithmic ramping/dimming
7:6	fader_src6	00b	R/W	LED6 Faders (see page 16)	
				00	fader disabled
				01	use fader1 (see page 23)
				10	use fader2
				11	use fader3

Table 34. Driver_Setup7 Register

Addr: A6h		Driver_Setup7 Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
4:0	color_slope7	00h	R/W	LED7 RGB Color Correction (see page 16) slope	
				00h	RGB Color Correction disabled
				01h	+0.15%/ºC
				...	...
				0Fh	+2.263%/ºC
				11h	-2.263%/ºC
				...	...
				1Fh	-0.15%/ºC
5	loglin7	1b	R/W	LED7 Logarithmic/Linear Ramping (see page 16)	
				0	linear ramping/dimming
				1	logarithmic ramping/dimming
7:6	fader_src7	00b	R/W	LED7 Faders (see page 16)	
				00	fader disabled
				01	use fader1 (see page 23)
				10	use fader2
				11	use fader3

Table 35. Driver_Setup8 Register

Addr: A7h		Driver_Setup8 Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
4:0	color_slope8	00h	R/W	LED8 RGB Color Correction (see page 16) slope	
				00h	RGB Color Correction disabled
				01h	+0.15%/ºC
				...	...
				0Fh	+2.263%/ºC
				11h	-2.263%/ºC
				...	...
				1Fh	-0.15%/ºC
5	loglin8	1b	R/W	LED8 Logarithmic/Linear Ramping (see page 16)	
				0	linear ramping/dimming
				1	logarithmic ramping/dimming
7:6	fader_src8	00b	R/W	LED8 Faders (see page 16)	
				00	fader disabled
				01	use fader1 (see page 23)
				10	use fader2
				11	use fader3

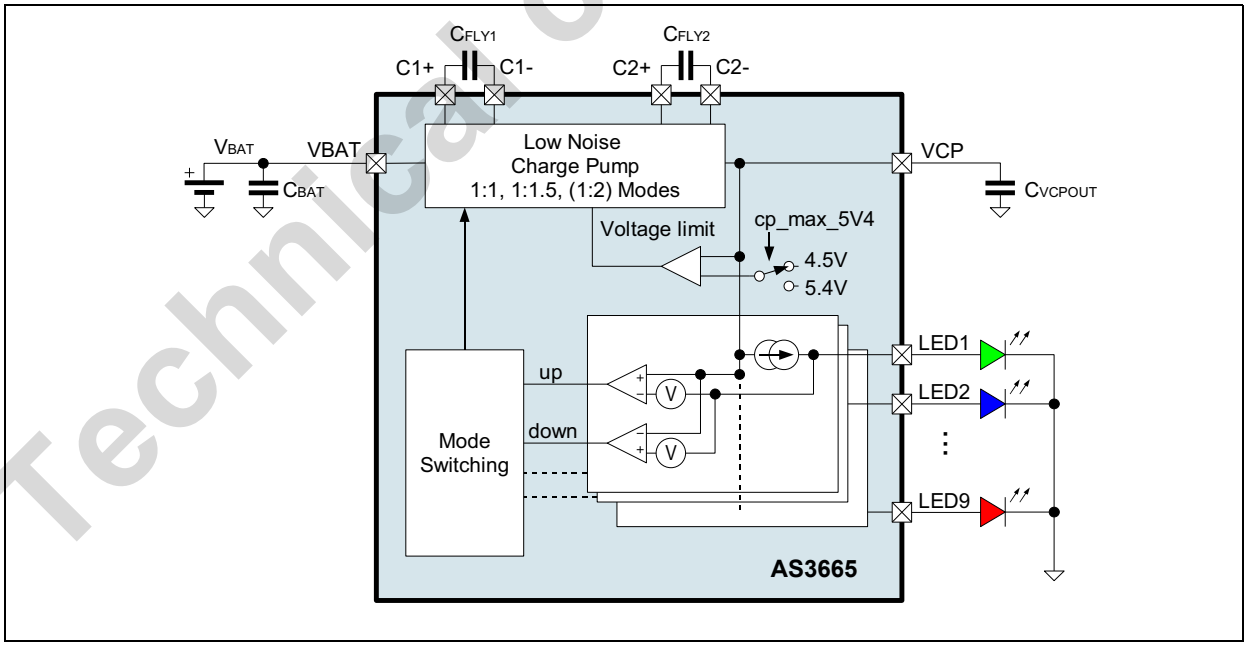
Table 36. Driver_Setup9 Register

Addr: A8h		Driver_Setup9 Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
4:0	color_slope9	00h	R/W	LED9 RGB Color Correction (see page 16) slope	
				00h	RGB Color Correction disabled
				01h	+0.15%/ºC
				...	...
				0Fh	+2.263%/ºC
				11h	-2.263%/ºC
				...	...
				1Fh	-0.15%/ºC
5	loglin9	1b	R/W	LED9 Logarithmic/Linear Ramping (see page 16)	
				0	linear ramping/dimming
				1	logarithmic ramping/dimming
7:6	fader_src9	00b	R/W	LED9 Faders (see page 16)	
				00	fader disabled
				01	use fader1 (see page 23)
				10	use fader2
				11	use fader3

Charge Pump

The charge pump used the two flying capacitors CFLY1 and CFLY2 to operate in 1:1, 1:1.5 and 1:2 mode boosting the input supply VBAT to VOUT (shown in Figure 19). An implemented soft start mechanism reduces the inrush current. Battery current is smoothed when switching the charge pump on and also at each switching condition. This precaution reduces electromagnetic radiation significantly.

Figure 19. Charge Pump



The operating modes are controlled according to the following tables:

Table 37. CP_Control Register

Addr: 05h		CP_Control Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
1:0	cp_mode	00b	R/W	Operating mode of charge pump (in manual mode sets the operating mode, in automatic mode reports the mode)	
				00	1:1 mode
				01	1:1.5 mode
				10	1:2 mode
				11	reserved - don't use
3:2	cp_mode_switching	00b	R/W	Mode switching control	
				00	1:1, 1:1.5 automatically up and down switching
				01	1:1, 1:1.5 automatically up switching
				10	1:1, 1:1.5, 1:2 automatically up switching
				11	Manual mode switching; mode defined by cp_mode
4	cp_auto_on	1b	R/W	Automatically switch on the charge pump if required	
				0	Charge pump should be enabled by cp_on
				1	CP is automatically enabled if a current source is enabled[1]
5	cp_on	0b	R/W	Automatically switch on the charge pump if required	
				0	The charge pump stays in 1:1 mode (unless cp_auto_on is set)
				1	Enable manual or automatic mode switching
7:6	cp_down_hyst	00b	R/W	Control the hysteresis for down switching from 1:1.5 to 1:1 mode	
				00	default hysteresis
				01	default-75mV hysteresis
				10	default-150mV hysteresis
				11	default-225mV hysteresis

1. Exception: LED7...LED9 if connected to VBAT. Defined by register LED7_on_cp, LED8_on_cp and LED9_on_cp.

The charge pump starts operation always in 1:1 mode and returns back to 1:1 mode if all current sources are switched off[7]. If the voltage across a enabled current source is no longer sufficient to operate the current source, the charge pump automatically select the next operating mode (which modes are allowed is controlled by cp_mode_switching. cp_auto_on or cp_on should be set for enabling this logic). In 1:1.5 mode and if cp_mode_switching=00, the charge pump also can automatically switch back into 1:1 mode if the voltage across all current sources is sufficiently high to use the more efficient 1:1 mode (a fine adjustment of this hysteresis is possible with cp_down_hyst).

Table 38. CP_Mode_Switch Register

Addr: 06h		CP_Mode_Switch Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
0	LED7_on_cp	1b	R/W	Configure if LED7 is powered by charge pump	
				0	LED7 is powered by VBAT (e.g. red LED)
				1	LED7 is powered from VOUT
1	LED8_on_cp	1b	R/W	Configure if LED8 is powered by charge pump	
				0	LED8 is powered by VBAT (e.g. red LED)
				1	LED8 is powered from VOUT
2	LED9_on_cp	1b	R/W	Configure if LED9 is powered by charge pump	
				0	LED9 is powered by VBAT (e.g. red LED)
				1	LED9 is powered from VOUT
3	cp_max_5V4	0b	R/W	Adjusts the maximum output voltage of the charge pump	
				0	charge pump VOUT regulates to 4.5V
				1	charge pump VOUT regulates to maximum 5.4V
4	cp_skip_on	1b	R/W	Allows pulse skip mode of charge pump	
				0	Pulse skip of charge pump is disabled
				1	Enable pulse skip of charge pump in low load conditions (reduce quiescent current in 1:1.5 mode)
5	cp_auto_reset	1b	R/W	If all current sources are off, reset the charge pump back to 1:1 mode	
				0	charge pump keeps last mode
				1	Reset charge pump to 1:1 if all current sources are off

Application Hint

Its usually safe to keep the default values of the charge pump registers. Only if a red LED is used (on LED7...LED9), reset the register bits LED7_on_cp=0, LED8_on_cp=0 and/or LED9_on_cp=0 to improve efficiency.

7. Exception: The manual mode switching mode (cp_mode_switching=11) can override this behavior.

General Purpose Output

The general purpose output ball can be used as an open drain PWM output pad, an ADC input or as a general purpose open drain output.

Table 39. LED_Control2 Register

Addr: 03h		LED_Control2 Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
7	GPO_on	0b	R/W	Enable PWM generator driving GPO	
				0	GPO PWM generator is off
				1	GPO PWM generator is enabled

The output pad GPO is controlled by register GPO_Control:

Table 40. GPO_Control Register

Addr: 04h		GPO_Control Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
1:0	gpo_mode	00b	R/W	Define operating mode of GPO ball	
				00	open drain PWM output
				01	open drain output of signal gpo_signal
				10	don't use
				11	
2	gpo_signal	0b	R/W	Status of GPO ball if gpo_mode=01	
				0	active low
				1	tristate or if used for ADC

Analog to Digital Converter

The AS3665 has a built-in 10-bit successive approximation analog-to-digital converter (ADC). It is internally supplied by C_{2V5}, which is also the full-scale input range (0V defines the ADC zero-code). For input signal exceeding C_{2V5} (typ. 2.5V) a resistor divider is used to scale the input of the ADC converter.

Table 41 shows the resolution and input ranges.

Table 41. ADC Input Ranges

Channel	Pin or Signal	Input Range	V_{LSB}	Note
0h	pin INT/AUDIO_IN if used with audio buffer	0.0V - 2.5V	NA	see section Audio Input on page 34
1h	junction temperature $ADC_{TEMPCODE}$	-30°C - 125°C	ADC_{TC}	see EQ 1
3h-5h	INT/AUDIO_IN, GPO, VBAT	0.0V - VBAT	ADC_{LSB}	internal voltage divider
6h-Fh	VOUT, LED1, LED2...LED9	0.0V - VOUT	ADC_{LSB}	internal voltage divider

The junction temperature can be calculated according to following formula ($ADC_{TEMPCODE}$ is the result of the ADC conversion from channel 1h):

$$T_{JUNCTION}\,[°C] = ADC_{TOFFSET} - ADC_{TC} * ADC_{TEMPCODE} \qquad (EQ\ 1)$$

The ADC is controlled by:

Table 42. ADC_Control Register

Addr: 09h		ADC_Control Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
3:0	adc_select	0h	R/W	Select ADC channel to be converted	
				0h	Audio Buffer (uses pin INT/AUDIO_IN and audio input amplifier - see section Audio Input on page 34)
				1h	$ADC_{TEMPCODE}$[1]
				2h	don't use
				3h	INT/AUDIO_IN
				4h	GPO[2]
				5h	VBAT
				6h	VOUT
				7h	LED1
				8h	LED2
				9h	LED3
				Ah	LED4
				Bh	LED5
				Ch	LED6
				Dh	LED7
				Eh	LED8
				Fh	LED9
5	adc_continuous	1b	R/W	Enable ADC continuous conversion	
				0	no continuous conversion
				1	ADC is continuously converting. If a conversion is finished an interrupt can be sent (register bit adc_eoc on page 37)
6	adc_slow	1b	R/W	select ADC conversion time	
				0	16µs ADC conversion time
				1	32µs ADC conversion time
7	adc_single_conversion	0b	W	writing '1' starts a single ADC conversion. If a conversion is finished an interrupt can be sent (register bit adc_eoc)	

1. Set temp_sens_on (see page 24)=1 before the measurement
2. set gpo_signal=1 and gpo_mode=01 to switch pad GPO into tristate

The ADC result is stored in registers adc<9:3> and adc<2:0>; a running conversion is identified by result_not_ready:

Table 43. ADC_MSB_Result Register

Addr: 0Ah		ADC_MSB_Result Register		
Bit	**Bit Name**	**Default**	**Access**	**Description**
6:0	adc<9:3>	NA	R	ADC Result bits 9:3 (MSBs)

Table 43. ADC_MSB_Result Register (Continued)

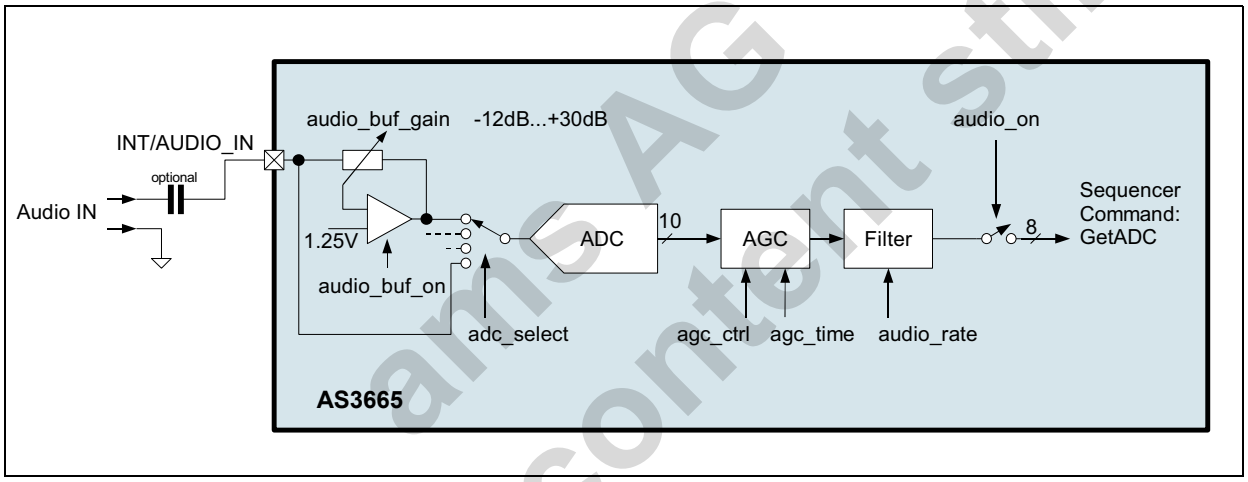
Addr: 0Ah		ADC_MSB_Result Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
7	result_not_ready	NA	R	Indicates end of ADC conversion cycle	
				0	Result is ready
				1	Conversion is running

Table 44. ADC_LSB_Result Register

Addr: 0Bh		ADC_LSB_Result Register		
Bit	**Bit Name**	**Default**	**Access**	**Description**
2:0	adc<2:0>	NA	R	ADC Result bits 2:0 (LSBs)

Audio Input

Figure 20. Audio Input internal Circuit



The audio input can be used to connect an analog audio signal to the AS3665 and do lighting effects dependent on this input signal on pad INT/AUDIO_IN[8].

The audio processing path is shown in Figure 20: The audio signal is amplified by the input amplifier with an adjustable gain setting to allow different audio input levels. With the ADC the signal is converted into a digital 10 bits signal. After the AGC, the data is filtered and then can be used with the sequencer command Get ADC (see page 67). The sequencers can then run different filter and processing algorithms to obtain the lighting effects.

Table 45. Audio_Control Register

Addr: 1Ch		Audio_Control Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
0	audio_on	0b	R/W	Enable AGC and Peak Detect for audio processing	
				0	Get ADC gets ADC value directly
				1	Get ADC uses AGC and audio filter -recommended setting if a audio signal is connected to the AS3665

8. Set int_mode=01 (analog input for ball INT/AUDIO_IN) and set adc_select=0 (to select audio buffer)

Table 45. Audio_Control Register (Continued)

Addr: 1Ch		Audio_Control Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
1	audio_cmdset	0b	R/W	Modifies the behavior for over/underflow with the sequencer adder and subtract commands	
				0	A over/underflow rolls over
				1	The adder/subtract command saturate at zero and full scale[1]
2	audio_buf_on	0b	R/W	Enable audio input buffer	
				0	Off any selection of adc_select possible
				1	On adc_select=0 (audio buffer) mandatory
5:3	audio_buf_gain	000b	R/W	Audio input buffer gain setting	
				000	-12dB
				001	-6dB
				010	0dB
				011	+6dB
				100	+12dB
				101	+18dB
				110	+24dB
				111	+30dB
7:6	reserved	00b	R/W	reserved - always set to 00b	

1. For audio processing always set audio_cmdset=1

AGC (Automatic Gain Control)

The AGC is used to 'compress' the input signal and to attenuate very low input amplitude signals (this is performed to ensure no light output for low signals especially for noisy input signals).

The AGC monitors the input signal amplitude and filters this amplitude with a filter with a short attack time, but a long decay time (decay time depends on the register agc_ctrl). This amplitude measurement (represented by an integer value from 0 to 15; the decay time of this measurement is controlled by agc_time) is then used to amplify or attenuate the input signal with one of the following amplification ratios (output to input ratio) – the curve A, B, or C is selected depending on the register agc_ctrl:

Table 46. AGC gain curves

Input Amplitude	AGC gain		
	curve A	**curve B**	**curve C**
0	0.0	0.0	0.0
1	7.5	5.0	3.5
2	7.0	4.0	3.0
3	4.5	3.5	2.5
4	3.5	3.0	2.0
5	3.0	2.5	1.5
6	2.5	2.5	1.5

Table 46. AGC gain curves

Input Amplitude	AGC gain		
	curve A	curve B	curve C
7	2.0	2.0	1.5
8	2.0	2.0	1.5
9	1.5	2.0	1.5
10	1.5	1.5	1.0
11	1.5	1.5	1.0
12	1.0	1.5	1.0
13	1.0	1.0	1.0
14	1.0	1.0	1.0
15	1.0	1.0	1.0

Table 47. Audio_AGC Register

Addr: 1Dh		Audio_AGC Register			
Bit	Bit Name	Default	Access	Description	
2:0	agc_ctrl	000b	R/W	Control AGC transfer function	
				000	AGC off (bypass)
				001	attenuate low amplitude signals otherwise linear response (to remove e.g. noise)
				010	AGC curve A; slow decay of amplitude detection
				011	AGC curve A; fast decay of amplitude detection
				100	AGC curve B; slow decay of amplitude detection
				101	AGC curve B; fast decay of amplitude detection
				110	AGC curve C; slow decay of amplitude detection
				111	AGC curve C; fast decay of amplitude detection
4:3	agc_time	00b	R/W	AGC amplitude detection decay time; minimum duration from min. gain to max. gain	
				00	460ms
				01	920ms
				10	1840ms
				11	3670ms

Interrupt Generator

The interrupt generator can send interrupt signals to e.g. the application processor to identify e.g. the end of pattern or a special event. When a not masked interrupt (register Interrupt_Mask) is triggered the INT/AUDIO_IN[9] pin is pulled low until the interrupt is reset by the I^2C interface.

Interrupt are readout by the Interrupt_Status register; pending interrupts are reset by writing back '1' to the register bit in Interrupt_Status which should be reset:

Following procedure to readout the interrupt is recommended:

9. The output should be enabled by setting register int_mode=00 (open drain interrupt output)

1. Readout Register Interrupt_Status
2. Write back the readout value in (1) to Interrupt_Status - this automatically resets all readout interrupts (and no interrupts can be lost)

Table 48. Interrupt_Status Register

Addr: 0Ch		Interrupt_Status Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
0	int1	0	R/W	Sequencer 1 has triggered an interrupt see End/Interrupt command on page 54	
				0	No interrupt
				1	Interrupt pending
1	int2	0	R/W	Sequencer 2 has triggered an interrupt see End/Interrupt command on page 54	
				0	No interrupt
				1	Interrupt pending
2	int3	0	R/W	Sequencer 3 has triggered an interrupt see End/Interrupt command on page 54	
				0	No interrupt
				1	Interrupt pending
3	no_extclock_detected	0	R/W	Monitor external clock detection on pin CLK32K - see Clock Generation on page 14	
				0	External clock is ok or internal clock is selected
				1	External clock is selected and no external clock is detected
4	init_ready_int	0	R/W	see Device Operating Mode on page 12	
				0	Initialization of the internal data of AS3665 is ongoing
				1	Initialization of the AS3665 is finished
5	adc_eoc	0	R/W	ADC end of conversion - see Analog to Digital Converter on page 32	
				0	ADC not started or conversion ongoing
				1	ADC has finished a conversion
6	ov_temp	0	R/W	see Temperature Supervision on page 39	
				0	Temperature ok
				1	Overtemperature detected

Interrupts can be enabled / disabled individually by the Interrupt_Mask register (if an interrupt is masked, it will not pull-down the pin INT/AUDIO_IN):

Table 49. Interrupt_Mask Register

Addr: 0Dh		Interrupt_Mask Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
0	int1_masked	1	R/W	0	No Mask
				1	int1 is masked
1	int2_masked	1	R/W	0	No Mask
				1	int2 is masked

Table 49. Interrupt_Mask Register (Continued)

Addr: 0Dh		Interrupt_Mask Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
2	int3_masked	1	R/W	0	No Mask
				1	int3 is masked
3	no_extclock_detected_masked	1	R/W	0	No Mask
				1	no_extclock_detected is masked
4	init_ready_int_masked	1	R/W	0	No Mask
				1	init_ready_int is masked
5	adc_eoc_masked	1	R/W	0	No Mask
				1	adc_eoc is masked
6	ov_temp_masked	1	R/W	0	No Mask
				1	ov_temp is masked

The interrupt output pad INT/AUDIO_IN is controlled by register GPO_Control:

Table 50. GPO_Control Register

Addr: 04h		GPO_Control Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
4:3	int_mode	00b	R/W	Define operating mode of INT/AUDIO_IN ball	
				00	open drain output of interrupt status
				01	push/pull output of signal int_signal
				10	analog input - use for Audio Input (see page 34) or Analog to Digital Converter (see page 32)
				11	
5	int_signal	0b	R/W	Status of INT/AUDIO_IN ball if int_mode=01	
				0	active low
				1	active high (VBAT)
7	int_on_trig	0b	R/W	Interrupt output selection flag	
				0	Interrupt status is available on ball INT/AUDIO_IN (if int_mode=00)
				1	Interrupt status is available on ball TRIG[1]

1. Set int_on_trig=1 if the ball INT/AUDIO_IN is used for audio and/or ADC and an interrupt output is required; the ball TRIG is then used as the interrupt open drain output

Trigger pin TRIG

Trigger commands can be sent by the internal sequencers to any other sequencer and or to/from the pin TRIG using the sequencer command Trigger (see page 55). The pin TRIG is active low, requires and external pullup resistor and the input should be enabled by setting trig_input_on=1.

Table 51. Exec_Mode Register

Addr: 01h		Exec_Mode Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
7	trig_input_on	0b	R/W	Enable external trigger input on pin TRIG	
				0	External trigger disabled
				1	External trigger enabled

Sent external trigger commands are three 32.768kHz clock cycles (see Clock Generation on page 14) long and received external triggers shall be longer than two clock cycles. During sending of an external trigger, the TRIG input is blocked.

Note: If two AS3665 devices send an external trigger at the exactly same time, the trigger command might get lost. Therefore it is recommended that only one AS3665 in a system should send trigger command and all other devices only receive trigger commands.

It is recommend to configure trig_input_on before program execution as changing trig_input_on during program execution can set a trigger pulse to the program.

LED Test

To test the LED in the production line, force a test current through the to be tested LED. Measure the voltage on the LED (by setting adc_select (see page 33) to the LED channel LED1...LED9). If the voltage on the LED is within the specified parameters for the LED, the LED is working properly.

Temperature Supervision

The temperature supervision protect the AS3665 against overtemperature - in case of overtemperature the AS3665 is reset (and therefore the charge pump is set back to 1:1 mode and all current sources are switched off). It is recommended to leave the temperature supervision always enabled (register bit ov_temp_on, default on):

Table 52. Supervision Register

Addr: 08h		Supervision Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
0	ov_temp_on[1]	1h	R/W	Overtemperature protection	
				0	Overtemperature protection disabled
				1	Overtemperature protection enabled
1	ov_temp_status	0h	R/W	Overtemperature protection triggered	
				0	No overtemperature detected
				1	Overtemperature detected

1. Always leave ov_temp_on set.

I²C mode Serial Data Bus

The AS3665 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 AS3665 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 AS3665 works in both modes. Connections to the bus are made through the open-drain I/O lines SDA and SCL.

I²C Address selection

The slave address can be selected depending on the external resistor RADDR connected to the pin ADDR. The actual address for reading and writing is selected according to Table 53.

Table 53. I²C Address Selection

RADDR	I²C Address[1] for	
	Writing	Reading
> 320kΩ (leave RADDR open)	80h	81h
320kΩ	82h	83h
160kΩ	84h	85h
80kΩ	86h	87h
40kΩ	88h	89h
20kΩ	8Ah	8Bh
10kΩ	8Ch	8Dh
0kΩ (short to GND)	8Eh	8Fh

1. This I²C address has 8 bits and includes the R/W flag (LSB). If a 7 bits address is required, use the 7 MSBs.

The following bus protocol has been defined (Figure 21):

- 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.

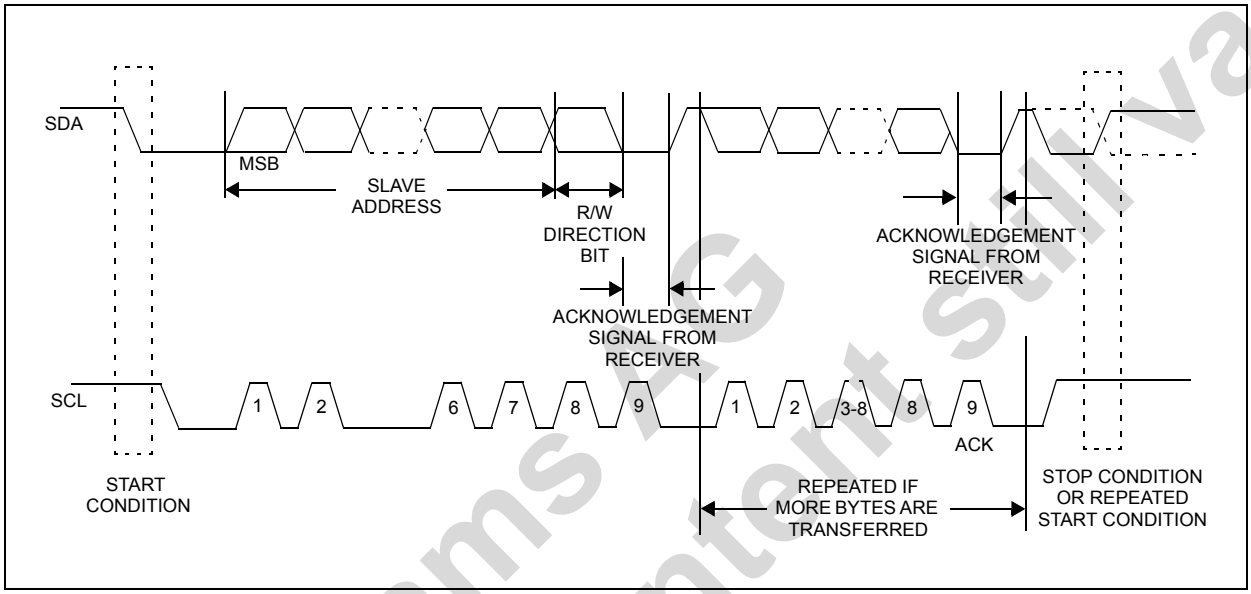
Each data transfer is initiated with a START condition and terminated with a STOP condition. The number of data bytes transferred between START and STOP conditions are not limited, and are determined by the master device. The information is transferred byte-wise and each receiver acknowledges with a 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 21. 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 AS3665 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 22). The slave address byte is the first byte received after the master generates the START condition. The slave address byte contains the 7-bit AS3665 address, which is 1000XXX[10], followed by the direction bit (R/W), which, for a write, is 0.[11] After receiving and decoding the slave address byte the device outputs an acknowledge on the SDA line. After the AS3665 acknowledges the slave address + write bit, the master transmits a register address to the AS3665. This sets the register pointer on the AS3665. The master may then transmit zero or more bytes of data (if more than one data byte is written

10.’XXX’ depends on the external resistor RADDR used; see I2C Address selection on page 40
11.The address for writing to the AS3665 is 8Xh = 1000XXX0b - see Table 53

see also Blockwrite/read boundaries on page 43), with the AS3665 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 AS3665 while the serial clock is input on SCL. START and STOP conditions are recognized as the beginning and end of a serial transfer (Figure 23 and Figure 24). The slave address byte is the first byte received after the master generates a START condition. The slave address byte contains the 7-bit AS3665 address, which is 1000XXX, followed by the direction bit (R/W), which, for a read, is 1.[12] After receiving and decoding the slave address byte the device outputs an acknowledge on the SDA line. The AS3665 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 also Blockwrite/read boundaries on page 43). 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 AS3665 must receive a “not acknowledge” to end a read.

Figure 22. Data Write - Slave Receiver Mode

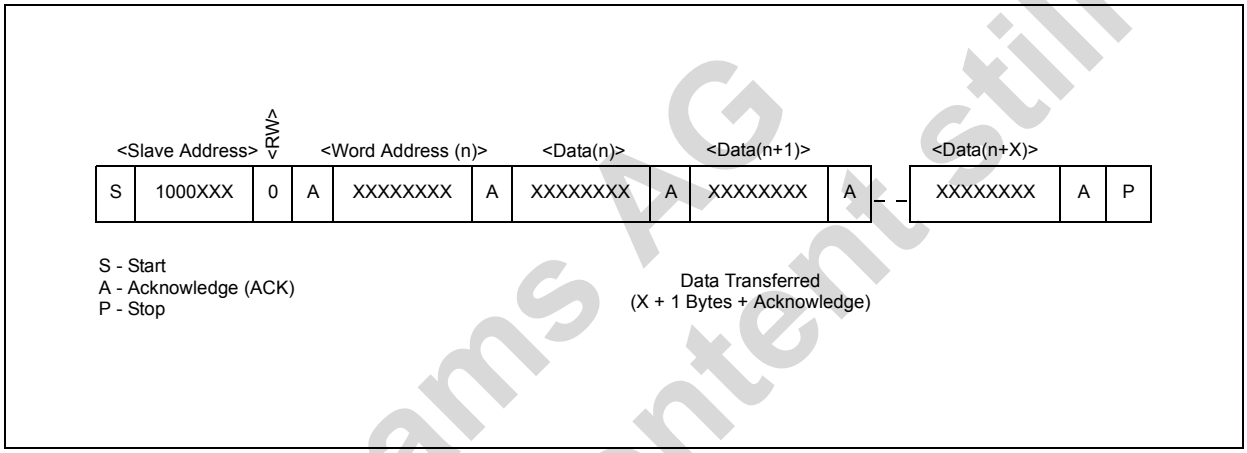
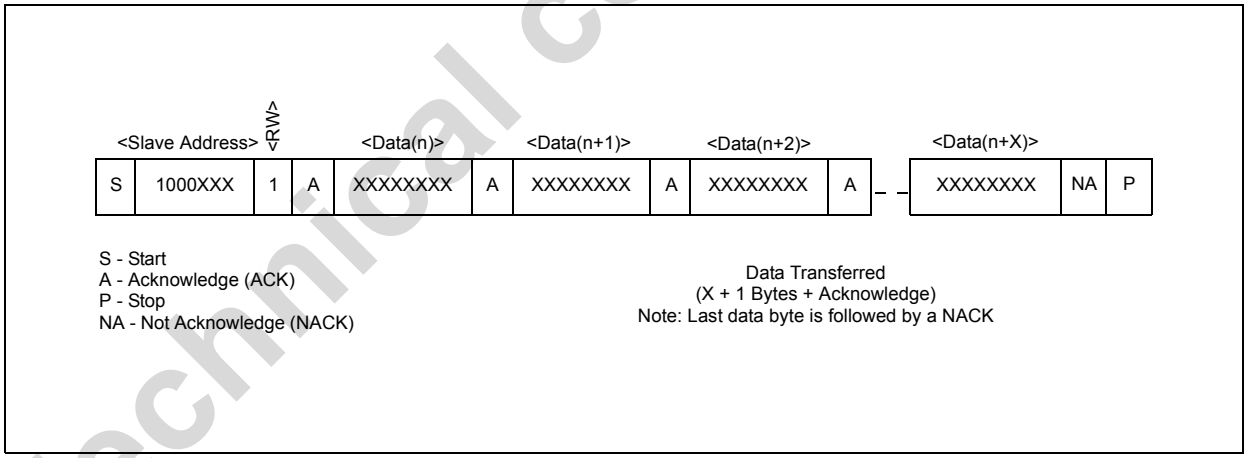
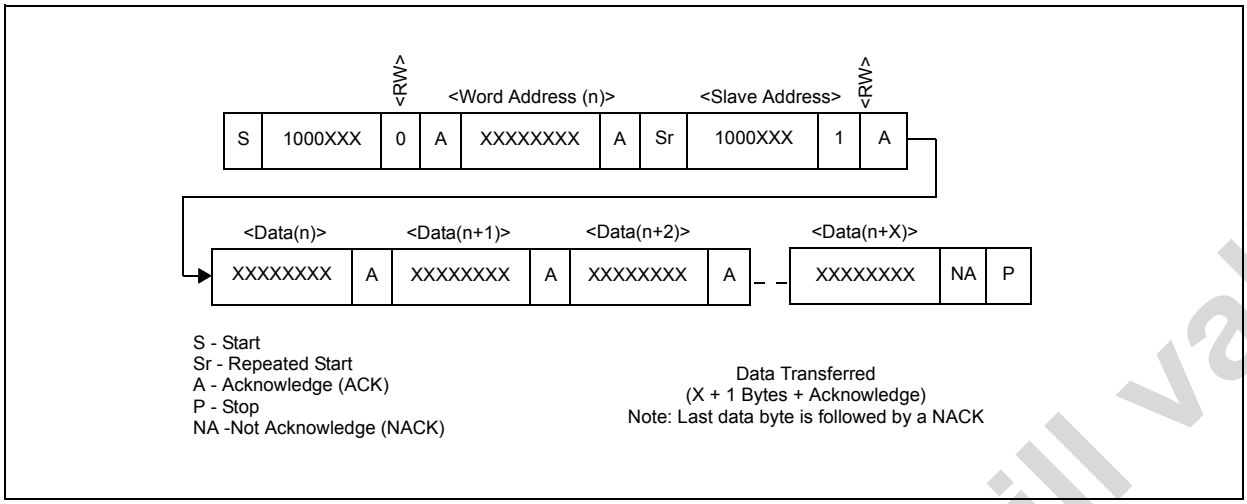


Figure 23. Data Read (from Current Pointer Location) - Slave Transmitter Mode



12. The address for read mode from the AS3665 is 8Xh+1 = 1000XXX1b - see Table 53

Figure 24. Data Read (Write Pointer, Then Read) - Slave Receive and Transmit



Blockwrite/read boundaries

If more than a single data-byte is written to or read from the AS3665 the address boundaries described in Table 54shall not be crossed[13]:

Table 54. Blockwrite/read boundaries

Area	Start	End
Area 1	00h	0Fh
Area 2	10h	18h
Area 3	19h	3Eh
Area 4 - Program Page Select	5Fh	
Area 5 - Program Access	60h	7Fh
Area 7	80h	CEh
Area 8 - SRAM	D0h	DFh
Area 9 - Program Direct Access	FEh - special I²C command	

Program Downloading

There are two possibilities to download programs - Program Direct Access and Program Download using Page Select[14]:

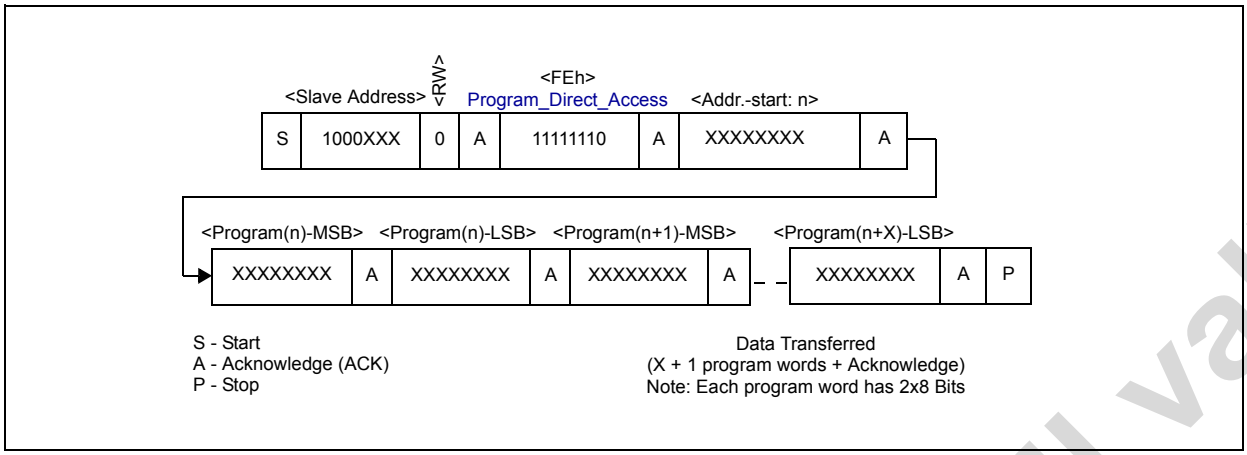
Program Direct Access

Wring to I²C register Program_Direct_Access allows direct access to the complete internal program memory using a single blockwrite command. Program downloading starts from address <n> and each program word is transferred with two I²C bytes (MSB first) as shown in Figure 25.

13.A single blockread or write shall not operate e.g. from 5Fh to 62h.

14.Choose the type of program download which fits best to the I²C controller

Figure 25. Program Write - Slave Receiver Mode



Program Download using Page Select

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...Cmd_F_MSB, Cmd_F_LSB (I²C registers area 60h to 7Fh)[15].

Table 55. Page_Select Register

Addr: 5Fh		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	don't use
				111	don't use

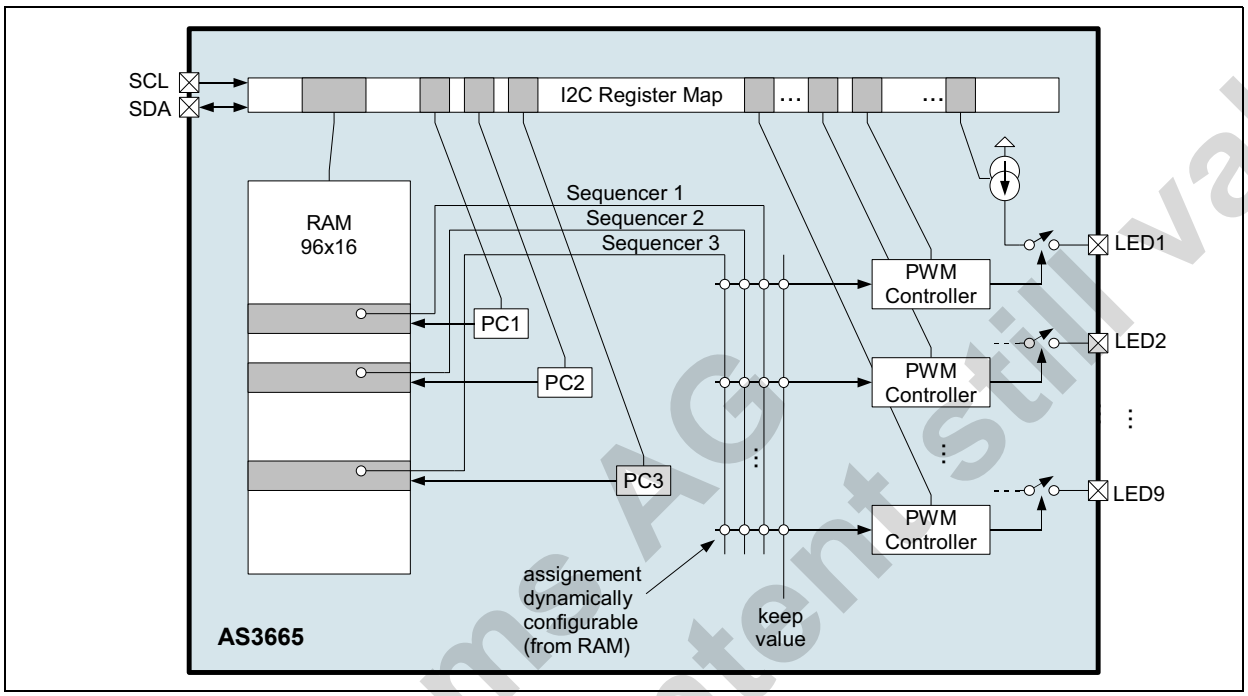
15. Setting page_select and writing of the program content shall use separate I²C commands (see Blockwrite/ read boundaries on page 43)

9 Programming

Concept

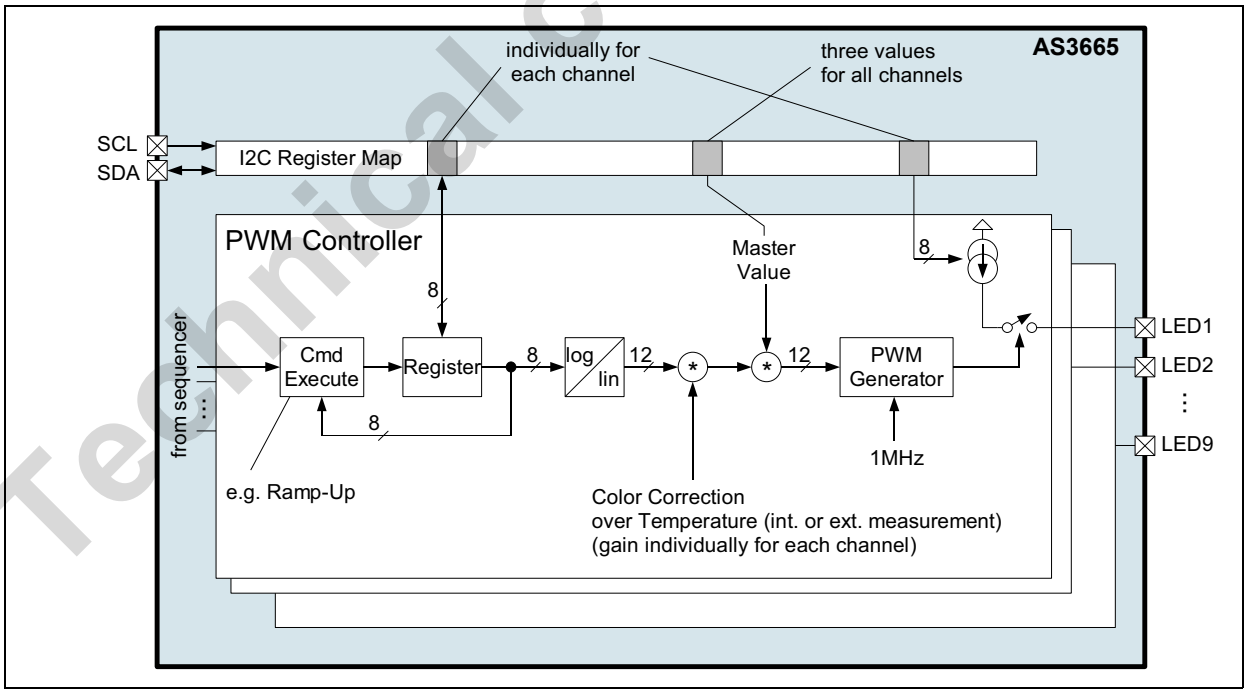
The internal structure for the sequencers, memory, PWM generator and I²C map is shown in Figure 26:

Figure 26. Internal Sequencers Structure



The AS3665 includes three program controlled sequencers operating on the internal memory. Each of these sequencers can be dynamically mapped to any of the PWM generator. Each of the PWM controllers has following structure:

Figure 27. PWM Controllers



It uses the command delivered by the sequencers, executes them, converts the data from linear to logarithmic representation add color correction and a master value. This signal is then feed into the actual PWM generator which controls the LED current source.

Program Execution and Debugging

Following steps are required for the setup of the AS3665 and execution of a program

1. The AS3665 operating mode should be standby or active - see Device Operating Mode on page 12
2. Set the LED currents - see Current Sources on page 15
3. The charge pump usually can be left at their default setting - see CP setting Application Hint on page 31
4. Download of program: see Program Downloading on page 43.
5. Write the program start addresses to registers start_addr1, start_addr2 and start_addr3[16]
6. Initialize the program counters PC1...PC3 by setting p1_en=01, p2_en=01 and p3_en=01. The program execution is automatically enabled (p1_en...p3_en is set to 10 by the AS3665).
7. Set AS3665 operating mode to active by setting chip_en=1 - see Device Operating Mode on page 12
8. Execute the program by setting p1_mode=10, p2_mode=10 and p3_mode=10

   Sequencers can be stopped by setting p1_mode...p3_mode=00 (hold). Single step debugging is achieved by setting p1_mode...p3_mode=01.[17] The program counter can be controller either by direct writing to registers PC1...PC3 or reset with p1_en...p3_en as shown above

9. Use AS3665 standby mode (set chip_en=0) to stop all programs and disable all current sources

Table 56. Exec_Enable Register

Addr: 00h		Exec_Enable Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
1:0	p1_en	00b	R/W	Execution enable for sequencer 1	
				00	Sequencer 1 is disabled[1]
				01	Reload program counter and enable: set PC1 to start_addr1, initialize sequencer 1 internal loop counters then set p1_en=10 (run)
				10	Execute sequencer commands as defined by p1_mode
				11	don't use
3:2	p2_en	00b	R/W	Execution enable for sequencer 2	
				00	Sequencer 2 is disabled[1]
				01	Reload program counter and enable: set PC2 to start_addr2, initialize sequencer 2 internal loop counters then set p2_en=10 (run)
				10	Execute sequencer commands as defined by p2_mode
				11	don't use

16. Assuming all three sequencers are actually used for the program.
17. The demoboard software simplifies the debugging using a graphical user interface.

Table 56. Exec_Enable Register (Continued) (Continued)

Addr: 00h		Exec_Enable Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
5:4	p3_en	00b	R/W	Execution enable for sequencer 3	
				00	Sequencer 3 is disabled[1]
				01	Reload program counter and enable: set PC3 to start_addr3, initialize sequencer 3 internal loop counters then set p3_en=10 (run)
				10	Execute sequencer commands as defined by p3_mode
				11	don't use

1. If all sequencers are switched off (p1_en=00, p2_en=00 and p3_en=00), LED1_on...LED9_on control the operation of the LEDs - see Current Sources on page 15

The Exec_Mode register defines the sequencer executing mode (e.g. single step or run):

Table 57. Exec_Mode Register

Addr: 01h		Exec_Mode Register			
Bit	**Bit Name**	**Default**	**Access**	**Description**	
1:0	p1_mode	00b	R/W	Execution mode for sequencer 1 if p1_en=10	
				00	Hold - finish current instruction and stop.
				01	Step - execute one instruction at PC1 and increment PC1 then reset p1_mode (hold)
				10	Run - start execution from PC1
				11	Step in place - execute one instruction at PC1 but don't increment PC1 then reset p1_mode (hold)
3:2	p2_mode	00b	R/W	Execution mode for sequencer 2 if p2_en=10	
				00	Hold - finish current instruction and stop.
				01	Step - execute one instruction at PC2 and increment PC2 then reset p2_mode (hold)
				10	Run - start execution from PC2
				11	Step in place - execute one instruction at PC2 but don't increment PC2 then reset p2_mode (hold)
5:4	p3_mode	00b	R/W	Execution mode for sequencer 3 if p3_en=10	
				00	Hold - finish current instruction and stop.
				01	Step - execute one instruction at PC3 and increment PC3 then reset p3_mode (hold)
				10	Run - start execution from PC3
				11	Step in place - execute one instruction at PC3 but don't increment PC3 then reset p3_mode (hold)

The program memory areas are setup using start_addr1...start_addr3:

Table 58. Start_Addr1 Register

Addr: B0h		Start_Addr1 Register		
Bit	**Bit Name**	**Default**	**Access**	**Description**
7:0	start_addr1	00h	R/W	Sequencer 1 start of program

Table 59. Start_Addr2 Register

Addr: B1h		Start_Addr2 Register		
Bit	**Bit Name**	**Default**	**Access**	**Description**
7:0	start_addr2	00h	R/W	Sequencer 2 start of program

Table 60. Start_Addr3 Register

Addr: B2h		Start_Addr3 Register		
Bit	**Bit Name**	**Default**	**Access**	**Description**
7:0	start_addr3	00h	R/W	Sequencer 3 start of program

The actual program execution of the sequencers is defined by the program counters PC1...PC3:

Table 61. Seq1_PC Register

Addr: B4h		Seq1_PC Register		
Bit	**Bit Name**	**Default**	**Access**	**Description**
7:0	PC1	00h	R/W	Sequencer 1 program counter

Table 62. Seq2_PC Register

Addr: B5h		Seq2_PC Register		
Bit	**Bit Name**	**Default**	**Access**	**Description**
7:0	PC2	00h	R/W	Sequencer 2 program counter

Table 63. Seq3_PC Register

Addr: B6h		Seq3_PC Register		
Bit	**Bit Name**	**Default**	**Access**	**Description**
7:0	PC3	00h	R/W	Sequencer 3 program counter

Sequencers

All three sequences are autonomous program execution unit executing the commands described in Sequencer Commands Table (see page 66). Programs are downloaded, started and stopped as described in Program Downloading (see page 43). The output of these sequencers is used for the PWM generator defined by so called MUX tables:

MUX tables - assignments of sequencers to channels

The MUX tables are setup during a program execution dynamically with the following sequencer commands:

- MUX set start address (see page 55) and MUX set end address (see page 56) define a memory region where the MUX tables is operating (MUX next address or MUX previous address). MUX set start address automatically loads the MUX for this sequencer with the content of the memory of 'start address'.
- MUX next address (see page 56) and MUX previous address (see page 57) increase (or decrease) the MUX pointer by one and load the MUX of this sequencer with the memory content the pointer is addressing. The MUX pointer is kept within range defined by MUX set start address and MUX set end address.
- MUX set ptr (see page 58) sets the MUX pointer to a address defined by a displacement and MUX set start address
- MUX select LED (see page 56) selects a single PWM output (single LED) where this sequencer is connected to. This is useful for simple sequencer - PWM connections without requiring to setup a dedicated MUX table.
- MUX clear (see page 56) clears the MUX of this sequencer (no PWM channels are selected anymore).

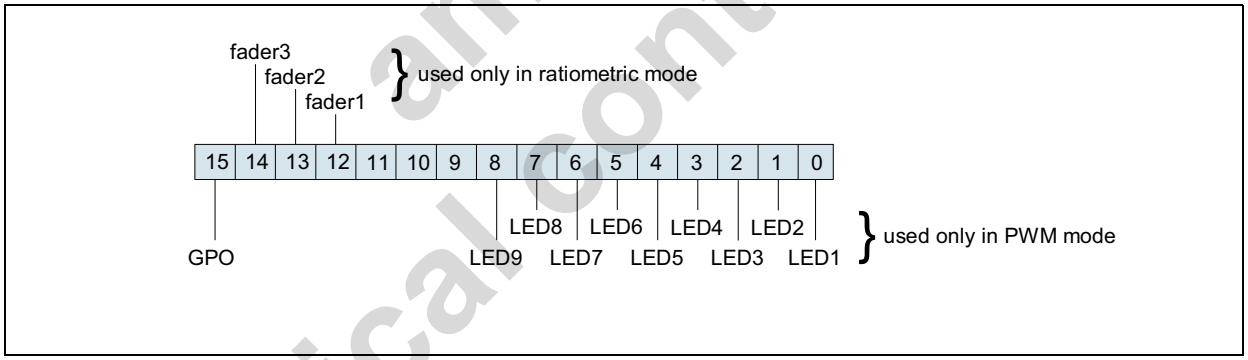
The sequencer can operate in two operating modes:

1. PWM mode - this is the standard operating mode; the sequencer directly controls any of the PWM generators. This is the default operating mode.
2. Ratiometric mode - the sequencer controls one or more of the faders (fader1 (see page 23), fader2 and/or fader3). The fader can control general LED brightness (configurable to control any number of LEDs) - see Current Sources (see page 15).[18]

The ratiometric mode is entered with the command MUX set RM (see page 66) or MUX fade (see page 66). The AS3665 returns to PWM mode with the command MUX reset RM.[19]

The sequencer are connected to the PWM generators and faders according to Figure 28 (the 16 bits are the content of the memory register, the MUX pointer is pointing to. A '1' connects the sequencer to this output, a '0' disconnects this output):

Figure 28. MUX table connections



Variables

The AS3665 includes four variables ra, rb, rc and rd. These variables can read and written by the I^2C interface and in parallel read and written by the sequencers[20]. Using the variables, programs can be controlled by a single I^2C commands. Sequencers can use these variables for internal calculations, for communication between the sequencers and to communicate to the I^2C controller.

18. The MUX tables share the same start address set by MUX set start address but have separate current addresses and end addresses set by MUX set end address
19. Use only the highest (in order 1,2,3) sequencers for ratiometric mode (e.g. SEQ1 PWM, SEQ2 ratiometric but not SEQ3 for PWM mode at the same time)
20. Variable rd is read/writable by I^2C but only readable by the sequencers.

There are two local variables (local to each sequencer): ra and rb - each sequencer sees its own variable:

Table 64. Variable_A1 Register

Addr: B8h		Variable_A1 Register		
Bit	**Bit Name**	**Default**	**Access**	**Description**
7:0	var_a1	00h	R/W	Sequencer 1 local variable ra

Table 65. Variable_A2 Register

Addr: B9h		Variable_A2 Register		
Bit	**Bit Name**	**Default**	**Access**	**Description**
7:0	var_a2	00h	R/W	Sequencer 2 local variable ra

Table 66. Variable_A3 Register

Addr: BAh		Variable_A3 Register		
Bit	**Bit Name**	**Default**	**Access**	**Description**
7:0	var_a3	00h	R/W	Sequencer 3 local variable ra

Table 67. Variable_B1 Register

Addr: BCh		Variable_B1 Register		
Bit	**Bit Name**	**Default**	**Access**	**Description**
7:0	var_b1	00h	R/W	Sequencer 1 local variable rb

Table 68. Variable_B2 Register

Addr: BDh		Variable_B2 Register		
Bit	**Bit Name**	**Default**	**Access**	**Description**
7:0	var_b2	00h	R/W	Sequencer 2 local variable rb

Table 69. Variable_B3 Register

Addr: BEh		Variable_B3 Register		
Bit	**Bit Name**	**Default**	**Access**	**Description**
7:0	var_b3	00h	R/W	Sequencer 3 local variable rb

There are two global variables: rc and rd - these are shared between all sequencers:

Table 70. Variable_C Register

Addr: BBh		Variable_C Register		
Bit	**Bit Name**	**Default**	**Access**	**Description**
7:0	var_c	00h	R/W	global variable rc - variable available for all sequencers

Table 71. Variable_D Register

Addr: 0Fh		Variable_D Register		
Bit	**Bit Name**	**Default**	**Access**	**Description**
7:0	var_d	00h	R/W	global variable rd - variable available for all sequencers

Audio Processing

Use austriamicrosystems sample codes for audio processing.

Sequencer Commands

Ramping of PWM(s) is achieved by the Ramp/Wait command shown in Table 72. The selected channels are chosen by MUX tables - assignments of sequencers to channels on page 49. This command also can be used to wait for a defined time in the program execution (if number of increments = 0).

Table 72. Ramp/Wait Command

Ramp/Wait Command **Ramps the PWM of the selected PWM generator up or down; if the number of increments is zero, it simply waits**				
Name	**Bits**	**Bitname**	**Parameter Description**	
Ramp/Wait	Compiler syntax: RMP, prescale, step time, sign, number of increments;			
	D15	0		
	D14	prescale	0	each step has 16 clock cycles (typ. 0.49ms at 32768Hz)
			1	each step has 512 clock cycles (typ. 15.6ms at 32768Hz)
			the clock generation is described in section Clock Generation on page 14	
	D13:D9	step time	1-31	duration between single increments/decrements e.g. if step time=8, prescale=0, sign=0, the duration between every increment is typically 0.49ms*8 = 3.92ms
	D8	sign	0	ramp up, always increment by 1; 255 is maximum value
			1	ramp down, always decrement by 1; 0 is minimum value
	D7:D0	number of increments	0	Wait for duration defined by prescale and step time
			1-255	number of actual cycles in a single ramp command (e.g. 255 defines a full scale ramp)

With the Set PWM command PWM(s) (PWM channels are connected to a sequencer as shown in section MUX tables - assignments of sequencers to channels on page 49) can be immediately forced to a value:

Table 73. Set PWM Command

Set PWM Command **Force PWM**				
Name	**Bits**	**Bitname**	**Parameter Description**	
Set PWM	Compiler syntax: SPW, pwm value;			
	D15:D8	01000000b (40h)		
	D7:D0	pwm value	0-255	actual PWM value used: 0...off 255...full scale

Ramping of PWM(s) dependent on variables is achieved by the Ramp with variable command shown in Table 74. This command also can be used to wait for a defined time in the program execution (if number of increments = 0).

Table 74. Ramp with variable Command

Ramp with variable Command **Ramps the PWM of the selected PWM generator up or down; if the number of increments is zero, it simply waits**				
Name	**Bits**	**Bitname**	**Parameter Description**	
Ramp with variable	Compiler syntax: RWV, prescale, sign, variable for step, variable for number of increments;			
	D15:D6	10000100_00b		
	D5	prescale	0	each step has 16 clock cycles (typ. 0.49ms at 32768Hz)
			1	each step has 512 clock cycles (typ. 15.6ms at 32768Hz)
			the clock generation is described in section Clock Generation on page 14	
	D4	sign	0	ramp up, always increment by 1; 255 is maximum value
			1	ramp down, always decrement by 1; 0 is minimum value
	D3:D2	variable for step	The content of the variable defines the duration between single increments/decrements; e.g. if variable rx=8, prescale=0, sign=0, the duration between every increment is typically 0.49ms*8 = 3.92ms	
			0	variable ra
			1	variable rb
			2	variable rc
			3	variable rd
	D1:D0	variable for number of increments	If the content of the variable rx is 0 then wait for duration defined by prescale and D3:D2 1-255 then it defines the number of actual cycles in a single ramp command (e.g. 255 defines a full scale ramp)	
			0	variable ra
			1	variable rb
			2	variable rc
			3	variable rd

With the Set PWM to variable command PWM(s) (PWM channels are connected to a sequencer as shown in section MUX tables - assignments of sequencers to channels on page 49) can be immediately forced to a value of a variable:

Table 75. Set PWM to variable Command

Set PWM to variable Command Force PWM				
Name	**Bits**	**Bitname**	**Parameter Description**	
Set PWM to variable	Compiler syntax: SPV, variable;			
	D15:D2	10000100_011000b		
	D1:D0	variable	The content of the variable is used to set the PWM value: 0...off 255...full scale	
			0	variable ra
			1	variable rb
			2	variable rc
			3	variable rd

With the GoTo Start command the program counter of the sequencer is reset to its start value:

Table 76. GoTo Start Command

GoTo Start Command			
Name	**Bits**	**Bitname**	**Parameter Description**
GoTo Start	Compiler syntax: GTS;		
	D15:D0	00000000_00000000b (0000h)	Set sequencer program counter to start address if sequencer 1 then PC1 = start_addr1 if sequencer 2 then PC2 = start_addr2 if sequencer 3 then PC3 = start_addr3

With the Branch command loops can be implemented. Loops can be nested without limits:

Table 77. Branch Command

Branch Command				
Name	**Bits**	**Bitname**	**Parameter Description**	
Branch	Compiler syntax: BRN, loop count, step number;			
	D15:D13	010b		
	D12:D7	loop count	0	infinite loops
			1-63	1 to 63 loops
	D6:D0	step number	0-127	jump to ‘step number’ for ‘loop count’ times; sets the PC of this sequencer = ‘step number’; in the compiler ‘step number’ can be defined by a label

With the Branch with variable command loops can be implemented. The number of loops are defined by a variable.

Loops can be nested without limits:

Table 78. Branch with variable Command

Branch with variable Command				
Name	**Bits**	**Bitname**	**Parameter Description**	
Branch with variable	Compiler syntax: BRV, step number, variable;			
	D15:D9	1000011b		
	D8:D2	step number	0-127	jump to 'step number' for 'variable' times; sets the PC of this sequencer = 'step number'; in the compiler 'step number' can be defined by a label
	D1:D0	variable	The content of the variable defines the number of loops performed (0=infinite)	
			0	variable ra
			1	variable rb
			2	variable rc
			3	variable rd

With the End/Interrupt command command program execution is stopped and optionally an interrupt is sent:

Table 79. End/Interrupt command Command

End/Interrupt command Command				
Name	**Bits**	**Bitname**	**Parameter Description**	
End/Interrupt command	Compiler syntax: END, int, reset;			
	D15:D13	101b		
	D12	int	0	no interrupt is sent
			1	send an interrupt (see Interrupt Generator on page 36) and disable this sequencer e.g. for sequencer 1, int3=1 and p1_en (see page 46) = 00
	D11	reset	0	program counter is incremented by 1
			1	program counter is reset to start address e.g. for sequencer 1, PC1 = start_addr1
	D10:D0	000_00000000b	stop program execution by resetting px_mode e.g. for sequencer 1, p1_mode (see page 47) = 00	

With the Trigger command internal (between sequencers) and external (between several AS3665) synchronization is

possible (see Trigger pin TRIG on page 39):

Table 80. Trigger Command

Trigger Command				
Name	**Bits**	**Bitname**	**Parameter Description**	
Trigger	Compiler syntax: TRG, wait trigger channels, send trigger channels;			
	D15:D13	111b		
	Wait for trigger from...			
	D12	Ext Trig	0	no trigger
			1	wait for external trigger from pin TRIG[1]
	D11:D10	XXb		
	D9	CH3	0	no trigger
			1	wait for trigger from sequencer 3
	D8	CH2	0	no trigger
			1	wait for trigger from sequencer 2
	D7	CH1	0	no trigger
			1	wait for trigger from sequencer 1
	Send trigger to...			
	D6	Ext Trig	0	no trigger
			1	send trigger to pin TRIG
	D5:D4	XXb		
	D3	CH3	0	no trigger
			1	send trigger to sequencer 3
	D2	CH2	0	no trigger
			1	send trigger to sequencer 2
	D1	CH1	0	no trigger
			1	send trigger to sequencer 1
	D0	Xb		

1. Set trig_input_on (see page 39)=1 to enable the input.

With the MUX set start address and MUX set end address commands the memory area for the multiplexer between the sequencers and the output PWM generators are initialized. (see MUX tables - assignments of sequencers to channels on page 49):

Table 81. MUX set start address Command

MUX set start address Command				
Name	**Bits**	**Bitname**	**Parameter Description**	
MUX set start address	Compiler syntax: MSS, RAM address;			
	D15:D7	10011100 0b		
	D6:D0	RAM address	0-127	Sets the multiplexer start address to ‘RAM address’. After the next command is executed the multiplexer for this sequencer is initialized by the content of this ‘RAM address’.

A similar command is used to set the multiplexer memory area end address:

Table 82. MUX set end address Command

MUX set end address Command				
Name	**Bits**	**Bitname**	**Parameter Description**	
MUX set end address	Compiler syntax: MSE, RAM address;			
	D15:D7	10011100 1b		
	D6:D0	RAM address	0-127	Sets the multiplexer end address to 'RAM address'.

With the MUX select LED command the sequencer can be simply connected to a single output (if more than one output should be controlled by one sequencer see MUX tables - assignments of sequencers to channels (see page 49)):

Table 83. MUX select LED Command

MUX select LED Command				
Name	**Bits**	**Bitname**	**Parameter Description**	
MUX select LED	Compiler syntax: MSL, LED select;			
	D15:D7	10011101 0b		
	D6:D0	LED select	1-9	Connect this sequencer to a single output defined by 'LED select'; e.g. 3 selects output LED3

With the MUX clear command the multiplexer tables are initialized (see page 49):

Table 84. MUX clear Command

MUX clear Command			
Name	**Bits**	**Bitname**	**Parameter Description**
MUX clear	Compiler syntax: MCL;		
	D15:D0	10011101 00000000b (9D00h)	Clear the MUX table (this sequencer is not connected to any output)

With the MUX next address command the MUX pointer can be moved down in the MUX table (see page 49):

Table 85. MUX next address Command

MUX next address Command			
Name	**Bits**	**Bitname**	**Parameter Description**
MUX next address	Compiler syntax: MNA;		
	D15:D0	10011101 10000000b (9D80h)	increase the MUX pointer by one; if the address would be above the address defined by MUX set end address, reset the MUX pointer to the address defined by MUX set start address; load the MUX with the content of this memory address

With the MUX previous address command the MUX pointer can be moved up in the MUX table (see page 49):

Table 86. MUX previous address Command

MUX previous address Command				
Name	**Bits**	**Bitname**	**Parameter Description**	
MUX previous address	Compiler syntax: MPA;			
	D15:D0	10011101 11000000b (9D8Ch)	decrease the MUX pointer by one; if the address would be below the address defined by MUX set start address, reset the MUX pointer to the address defined by MUX set end address; load the MUX with the content of this memory address	

With the MUX set RM and MUX reset RM command the sequencer can be configured for ratiometric mode or PWM mode:

Table 87. MUX set RM Command

MUX set RM Command			
Name	**Bits**	**Bitname**	**Parameter Description**
MUX set RM	Compiler syntax: SRM;		
	D15:D0	10011101 00100000b (9D20h)	Set Sequencer ratiometric mode - see MUX tables - assignments of sequencers to channels on page 49

Table 88. MUX reset RM Command

MUX reset RM Command			
Name	**Bits**	**Bitname**	**Parameter Description**
MUX reset RM	Compiler syntax: RRM;		
	D15:D0	10011101 01000000b (9D40h)	Reset Sequencer ratiometric mode (= PWM mode) - see MUX tables - assignments of sequencers to channels on page 49

MUX fade is used to set the sequencer in ratiometric mode and configure the faders which are connected to this sequencer with one single command - no additional MUX tables are required:

Table 89. MUX fade Command

MUX fade Command				
Name	**Bits**	**Bitname**	**Parameter Description**	
MUX fade	Compiler syntax: MXF,<faders>;			
	D15:D3	10011101 00100b	Set Sequencer ratiometric mode - see MUX tables - assignments of sequencers to channels on page 49 and configure the faders, which are connected to this sequencer.	
	D2	fader3	1	sequencer controls fader 3
			0	sequencer does not control fader 3
	D1	fader2	1	sequencer controls fader 2
			0	sequencer does not control fader 2
	D0	fader1	1	sequencer controls fader 1
			0	sequencer does not control fader 1

MUX set ptr set the MUX pointer to an address <vector number>+MUX set start address:

Table 90. MUX set ptr Command

MUX set ptr Command			
Name	**Bits**	**Bitname**	**Parameter Description**
MUX set ptr	Compiler syntax: MXP,<vector number>;		
	D15:D5	10011101 011b	
	D4:D0	vector number	The MUX pointer is set to <vector number> + address defined by MUX set start address

With the je (jump ==), jge (jump >=), jl (jump <) and jne (jump <>) commands the program flow[21] can be controlled depending on values in variables:

Table 91. je (jump ==) Command

je (jump ==) Command				
Name	**Bits**	**Bitname**	**Parameter Description**	
je (jump ==)	Compiler syntax: JE, instructions skipped, variable 1, variable 2;			
	D15:D9	1000100b		
	D8:D4	instructions skipped	0-31	defines the number of instructions skipped, if variable1 = variable2 PC = PC + 'instructions skipped'
	D3:D2	variable 1	0	variable1 = ra
			1	variable1 = rb
			2	variable1 = rc
			3	variable1 = rd
	D1:D0	variable 2	0	variable2 = ra
			1	variable2 = rb
			2	variable2 = rc
			3	variable2 = rd

21. Only positive jumps (jump down) can be implemented. If jumps in both directions are required, use these commands in combination with Branch (see page 53)

Table 92. jge (jump >=) Command

jge (jump >=) Command				
Name	**Bits**	**Bitname**	**Parameter Description**	
jge (jump >=)	Compiler syntax: JGE, instructions skipped, variable 1, variable 2;			
	D15:D9	1000101b		
	D8:D4	instructions skipped	0-31	defines the number of instructions skipped, if variable1 >= variable2 PC = PC + 'instructions skipped'
	D3:D2	variable 1	0	variable1 = ra
			1	variable1 = rb
			2	variable1 = rc
			3	variable1 = rd
	D1:D0	variable 2	0	variable2 = ra
			1	variable2 = rb
			2	variable2 = rc
			3	variable2 = rd

Table 93. jl (jump <) Command

jl (jump <) Command				
Name	**Bits**	**Bitname**	**Parameter Description**	
jl (jump <)	Compiler syntax: JL, instructions skipped, variable 1, variable 2			
	D15:D9	1000110b		
	D8:D4	instructions skipped	0-31	defines the number of instructions skipped, if variable1 < variable2 PC = PC + 'instructions skipped'
	D3:D2	variable 1	0	variable1 = ra
			1	variable1 = rb
			2	variable1 = rc
			3	variable1 = rd
	D1:D0	variable 2	0	variable2 = ra
			1	variable2 = rb
			2	variable2 = rc
			3	variable2 = rd

Table 94. jne (jump <>) Command

jne (jump <>) Command				
Name	**Bits**	**Bitname**	**Parameter Description**	
jne (jump <>)	Compiler syntax: JNE, instructions skipped, variable 1, variable 2			
	D15:D9	1000111b		
	D8:D4	instructions skipped	0-31	defines the number of instructions skipped, if variable1 <> variable2 (not equal) PC = PC + 'instructions skipped'
	D3:D2	variable 1	0	variable1 = ra
			1	variable1 = rb
			2	variable1 = rc
			3	variable1 = rd
	D1:D0	variable 2	0	variable2 = ra
			1	variable2 = rb
			2	variable2 = rc
			3	variable2 = rd

Variable can be initialized to a constant value by the command ld (load):

Table 95. ld (load) Command

ld (load) Command				
Name	**Bits**	**Bitname**	**Parameter Description**	
ld (load)	Compiler syntax: LD target variable, value;			
	D15:D12	1001b (9h)		
	D11:D10	target variable	0	set ra = value
			1	set rb = value
			2	set rc = value
			3	don't use
	D9:D8	00b		
	D7:D0	value	0-255	value

A constant value can be added to a variable with the command add number:

Table 96. add number Command

add number Command				
Name	**Bits**	**Bitname**	**Parameter Description**	
add number	Compiler syntax: ADN, target variable, value;			
	D15:D12	1001b (9h)		
	D11:D10	target variable	0	set ra = ra + value
			1	set rb = rb + value
			2	set rc = rc + value
			3	don't use
	D9:D8	01b		
	D7:D0	value	0-255	value

Variable are added together with the command add variable:

Table 97. add variable Command

add variable Command				
Name	**Bits**	**Bitname**	**Parameter Description**	
add variable	Compiler syntax: ADV, target variable, variable 1, variable 2;			
	D15:D12	1001b (9h)		
	D11:D10	target variable	0	set ra = variable1 + variable2
			1	set rb = variable1 + variable2
			2	set rc = variable1 + variable2
			3	don't use
	D9:D4	110000b		
	D3:D2	variable 1	0	variable1 = ra
			1	variable1 = rb
			2	variable1 = rc
			3	variable1 = rd
	D1:D0	variable 2	0	variable2 = ra
			1	variable2 = rb
			2	variable2 = rc
			3	variable2 = rd

A constant value can be subtracted from a variable with the command sub number:

Table 98. sub number Command

sub number Command				
Name	**Bits**	**Bitname**	**Parameter Description**	
sub number	Compiler syntax: SBN, target variable, value;			
	D15:D12	1001b (9h)		
	D11:D10	target variable	0	set ra = ra - value
			1	set rb = rb - value
			2	set rc = rc - value
			3	don't use
	D9:D8	10b		
	D7:D0	value	0-255	value

Variable are subtracted with the command sub variable:

Table 99. sub variable Command

sub variable Command				
Name	**Bits**	**Bitname**	**Parameter Description**	
sub variable	Compiler syntax: SBV, target variable, variable 1, variable 2;			
	D15:D12	1001b (9h)		
	D11:D10	target variable	0	set ra = variable1 - variable2
			1	set rb = variable1 - variable2
			2	set rc = variable1 - variable2
			3	don't use
	D9:D4	110001b		
	D3:D2	variable 1	0	variable1 = ra
			1	variable1 = rb
			2	variable1 = rc
			3	variable1 = rd
	D1:D0	variable 2	0	variable2 = ra
			1	variable2 = rb
			2	variable2 = rc
			3	variable2 = rd

Audio Commands

austriamicrosystems provides audio programs to control light depending on an audio input as a starting point for an actual implementation. Due to the complexity of these programs it is recommend to use the demos and modify the demo codes accordingly.

With the command Get ADC, data can be fetched from the audio filter (See Audio Input on page 34):

Table 100. Get ADC Command

Get ADC Command				
Name	**Bits**	**Bitname**	**Parameter Description**	
Get ADC	Compiler syntax: GET, target variable;			
	D15:D4	10001010_0010b (8A2h)		
	D3:D0	target variable	0h	set ra = value from ADC or filter
			5h	set rb = value from ADC or filter
			Ah	set rc = value from ADC or filter
			Fh	set rd = value from ADC or filter
			other values	don't use

Memory Operation Command - load/store SRAM

Table 101. Load SRAM Command

Load SRAM Command				
Name	**Bits**	**Bitname**	**Parameter Description**	
Load SRAM	Compiler syntax: LDS, R/W, source/target variable;			
	D15:D9	1000_111b (87h)		
	D8	R/W	Load from or store to SRAM (Register SRAM0, SRAM1...SRAM15)	
			0	Read from SRAM: SRAM -> target variable
			1	Write to SRAM: source variable -> SRAM
	D7:D4	SRAM Address	Define SRAM address register to load from or store to	
			0	sram_0
			1	sram_1
			...	...
			F	sram_15
	D3:D0	source/target variable	Set source variable for read or target variable for write	
			0h	ra
			5h	rb
			Ah	rc
			Fh	rd
			other values	don't use

Logical Operation Commands

or command provides a binary or between variables:

Table 102. or Command

or Command				
Name	**Bits**	**Bitname**	**Parameter Description**	
or	Compiler syntax: OR, input variable, output variable;			
	D15:D9	1000101b		
	D8:D7	input variable	0	ra
			1	rb
			2	rc
			3	rd
	D6:D4	001b		
	D3:D0	output variable	0h	set ra = ra or <input variable>
			5h	set rb = rb or <input variable>
			Ah	set rc = rc or <input variable>
			Fh	set rd = rd or <input variable>
			other values	don't use

and command provides a binary and between variables:

Table 103. and Command

and Command				
Name	**Bits**	**Bitname**	**Parameter Description**	
and	Compiler syntax: AND, input variable, output variable;			
	D15:D9	1000110b		
	D8:D7	input variable	0	ra
			1	rb
			2	rc
			3	rd
	D6:D4	001b		
	D3:D0	output variable	0h	set ra = ra and <input variable>
			5h	set rb = rb and <input variable>
			Ah	set rc = rc and <input variable>
			Fh	set rd = rd and <input variable>
			other values	don't use

Shift Commands

shift left Variable shift a variable left by 1 (multiply by 2) - if the result exceeds 255, 255 is used as result:

Table 104. shift left Command

shift left Command				
Name	**Bits**	**Bitname**	**Parameter Description**	
shift left	Compiler syntax: SL, input variable, output variable;			
	D15:D9	1000101b		
	D8:D7	input variable	0	ra
			1	rb
			2	rc
			3	rd
	D6:D4	000b		
	D3:D0	output variable	0h	set ra = <input variable> * 2
			5h	set rb = <input variable> * 2
			Ah	set rc = <input variable> * 2
			Fh	set rd = <input variable> * 2
			other values	don't use

shift right Variable shifts a variable right by 1 (divide by 2, rounded to 0):

Table 105. shift right Command

shift right Command				
Name	**Bits**	**Bitname**	**Parameter Description**	
shift right	Compiler syntax: SR, input variable, output variable;			
	D15:D9	1000110b		
	D8:D7	input variable	0	ra
			1	rb
			2	rc
			3	rd
	D6:D4	000b		
	D3:D0	output variable	0h	set ra = <input variable> / 2
			5h	set rb = <input variable> / 2
			Ah	set rc = <input variable> / 2
			Fh	set rd = <input variable> / 2
			other values	don't use

10 Sequencer Commands Table

Table 106. Sequencer Commands Table

Command	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	see page
Ramp/Wait	0	prescale	step time					sign	number of increments								51
Set PWM	0	1	0	0	0	0	0	0	pwm value								51
Ramp with variable	1	0	0	0	0	1	0	0	0	0	prescale	sign	variable for step		variable increment		52
Set PWM to variable	1	0	0	0	0	1	0	0	0	1	1	0	0	0	variable		53
GoTo Start	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	53
Branch	1	0	1	loop count					step number								53
Branch with variable	1	0	0	0	0	1	1	step number							variable		54
End/Interrupt command	1	1	0	int	reset	0	0	0	0	0	0	0	0	0	0	0	54
Trigger	1	1	1	Wait for trigger from...						Send Trigger to...						X	55
				Ext Trig	X	X	CH3	CH2	CH1	Ext Trig	X	X	CH3	CH2	CH1		
MUX set start address	1	0	0	1	1	1	0	0	0	RAM address							55
MUX set end address	1	0	0	1	1	1	0	0	1	RAM address							56
MUX select LED	1	0	0	1	1	1	0	1	0	LED select							56
MUX clear	1	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	56
MUX next address	1	0	0	1	1	1	0	1	1	0	0	0	0	0	0	0	56
MUX previous address	1	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	57
MUX set RM	1	0	0	1	1	1	0	1	0	0	1	0	0	0	0	0	57
MUX reset RM	1	0	0	1	1	1	0	1	0	1	0	0	0	0	0	0	57
MUX fade	1	0	0	1	1	1	0	1	0	0	1	0	0	fader3	fader2	fader1	57
MUX set ptr	1	0	0	1	1	1	0	1	0	1	1	vector number					58
je (jump ==)	1	0	0	0	1	0	0	instructions skipped					variable 1		variable 2		58
jge (jump >=)	1	0	0	0	1	0	1	instructions skipped					variable 1		variable 2		59
jl (jump <)	1	0	0	0	1	1	0	instructions skipped					variable 1		variable 2		59
jne (jump <>)	1	0	0	0	1	1	1	instructions skipped					variable 1		variable 2		60
ld (load)	1	0	0	1	target variable		0	0	value								60
add number	1	0	0	1	target variable		0	1	value								61
add variable	1	0	0	1	target variable		1	1	0	0	0	0	variable 1		variable 2		61
sub number	1	0	0	1	target variable		1	0	value								62

Table 106. Sequencer Commands Table

Command	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	see page
sub variable	1	0	0	1	target variable		1	1	0	0	0	1	variable 1		variable 2		62
Get ADC	1	0	0	0	1	0	1	0	0	0	1	0	target variable				63
Load SRAM	1	0	0	0	1	1	1	R/W	SRAM Address				source/target variable				63
or	1	0	0	0	1	0	1	input variable		0	0	1	target variable				64
and	1	0	0	0	1	1	0	input variable		0	0	1	target variable				64
shift left	1	0	0	0	1	0	1	input variable		0	0	0	target variable				65
shift right	1	0	0	0	1	1	0	input variable		0	0	0	target variable				65

11 Registermap

Table 107. Register Map

Register Definition	Addr hex	Default	Content							
Name			**b7**	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**b0**
Exec_Enable	00h	00h	ram_init	chip_en	p3_en		p2_en		p1_en	
Exec_Mode	01h	00h	trig_input_on	0	p3_mode		p2_mode		p1_mode	
LED_Control1	02h	00h	LED8_on	LED7_on	LED6_on	LED5_on	LED4_on	LED3_on	LED2_on	LED1_on
LED_Control2	03h	00h	GPO_on	fader_log lin3	fader_log lin2	fader_log lin1	temp_comp_mode			LED9_on
GPO_Control	04h	00h	int_on_trig	sel_ext_clock	int_signal	int_mode		gpo_signal	gpo_mode	
CP_Control	05h	10h	cp_down_hyst		cp_on	cp_auto_on	cp_mode_switching		cp_mode	
CP_Mode_Switch	06h	37h			cp_auto_reset	cp_skip_on	cp_max_5V4	LED9_on_cp	LED8_on_cp	LED7_on_cp
Supervision	08h	81h	auto_shutdown		osc_always_on				ov_temp_status	ov_temp_on
ADC_Control	09h	00h	adc_single_conversion	adc_slow	adc_continuous	adc_select				
ADC_MSB_Result	0Ah	00h	result_not_ready	adc<9:3>						
ADC_LSB_Result	0Bh	00h						adc<2:0>		
Interrupt_Status	0Ch	40h		ov_temp	adc_eoc	init_ready_int	no_extclock_detected	int3	int2	int1
Interrupt_Mask	0Dh	FFh		ov_temp_masked	adc_eoc_masked	init_ready_int_masked	no_extclock_detected_masked	int3_masked	int2_masked	int1_masked
Temp_Sense_Control	0Eh	00h						temp_meas_busy	temp_sens_on	temp_int_ext
Variable_D	0Fh	00h	var_d							
LED_Current1	10h	00h	LED_current1							
LED_Current2	11h	00h	LED_current2							
LED_Current3	12h	00h	LED_current3							
LED_Current4	13h	00h	LED_current4							
LED_Current5	14h	00h	LED_current5							
LED_Current6	15h	00h	LED_current6							
LED_Current7	16h	00h	LED_current7							
LED_Current8	17h	00h	LED_current8							
LED_Current9	18h	00h	LED_current9							
LED_MaxCurr1	19h	00h	LED4_max		LED3_max		LED2_max		LED1_max	

Table 107. Register Map (Continued)

Register Definition	Addr hex	Default	Content							
Name			**b7**	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**b0**
LED_MaxCurr2	1Ah	00h	LED8_max		LED7_max		LED6_max		LED5_max	
LED_MaxCurr3	1Bh	00h							LED9_max	
Audio_Control	1Ch	00h	0	0	audio_buf_gain			audio_buf_on	audio_cmdset	audio_on
Audio_AGC	1Dh	00h				agc_time		agc_ctrl		
LED_Temp	1Fh	00h	led_temp							
Reset_Control	3Ch	00h								force_reset
Chip_ID1	3Dh	C9h	1	1	0	0	1	0	0	1
Chip_ID2	3Eh	5xh	0	1	0	1	revision			
Page_Select	5Fh	00h						page_select		
Cmd_0_MSB	60h	00h	cmd_0_msb							
Cmd_0_LSB	61h	00h	cmd_0_lsb							
Cmd_1_MSB	62h	00h	cmd_1_msb							
Cmd_1_LSB	63h	00h	cmd_1_lsb							
Cmd_2_MSB	64h	00h	cmd_2_msb							
Cmd_2_LSB	65h	00h	cmd_2_lsb							
Cmd_3_MSB	66h	00h	cmd_3_msb							
Cmd_3_LSB	67h	00h	cmd_3_lsb							
Cmd_4_MSB	68h	00h	cmd_4_msb							
Cmd_4_LSB	69h	00h	cmd_4_lsb							
Cmd_5_MSB	6Ah	00h	cmd_5_msb							
Cmd_5_LSB	6Bh	00h	cmd_5_lsb							
Cmd_6_MSB	6Ch	00h	cmd_6_msb							
Cmd_6_LSB	6Dh	00h	cmd_6_lsb							
Cmd_7_MSB	6Eh	00h	cmd_7_msb							
Cmd_7_LSB	6Fh	00h	cmd_7_lsb							
Cmd_8_MSB	70h	00h	cmd_8_msb							
Cmd_8_LSB	71h	00h	cmd_8_lsb							
Cmd_9_MSB	72h	00h	cmd_9_msb							
Cmd_9_LSB	73h	00h	cmd_9_lsb							
Cmd_A_MSB	74h	00h	cmd_A_msb							
Cmd_A_LSB	75h	00h	cmd_A_lsb							
Cmd_B_MSB	76h	00h	cmd_B_msb							
Cmd_B_LSB	77h	00h	cmd_B_lsb							
Cmd_C_MSB	78h	00h	cmd_C_msb							

Table 107. Register Map (Continued)

Register Definition	Addr hex	Default	Content							
Name			b7	b6	b5	b4	b3	b2	b1	b0
Cmd_C_LSB	79h	00h	cmd_C_lsb							
Cmd_D_MSB	7Ah	00h	cmd_D_msb							
Cmd_D_LSB	7Bh	00h	cmd_D_lsb							
Cmd_E_MSB	7Ch	00h	cmd_E_msb							
Cmd_E_LSB	7Dh	00h	cmd_E_lsb							
Cmd_F_MSB	7Eh	00h	cmd_F_msb							
Cmd_F_LSB	7Fh	00h	cmd_F_lsb							
PWM_LED1	80h	00h	pwm_LED1							
PWM_LED2	81h	00h	pwm_LED2							
PWM_LED3	82h	00h	pwm_LED3							
PWM_LED4	83h	00h	pwm_LED4							
PWM_LED5	84h	00h	pwm_LED5							
PWM_LED6	85h	00h	pwm_LED6							
PWM_LED7	86h	00h	pwm_LED7							
PWM_LED8	87h	00h	pwm_LED8							
PWM_LED9	88h	00h	pwm_LED9							
PWM_GPO	8Fh	00h	pwm_GPO							
Fader1	9Bh	00h	fader1							
Fader2	9Ch	00h	fader2							
Fader3	9Dh	00h	fader3							
Driver_Setup1	A0h	20h	fader_src1		loglin1	color_slope1				
Driver_Setup2	A1h	20h	fader_src2		loglin2	color_slope2				
Driver_Setup3	A2h	20h	fader_src3		loglin3	color_slope3				
Driver_Setup4	A3h	20h	fader_src4		loglin4	color_slope4				
Driver_Setup5	A4h	20h	fader_src5		loglin5	color_slope5				
Driver_Setup6	A5h	20h	fader_src6		loglin6	color_slope6				
Driver_Setup7	A6h	20h	fader_src7		loglin7	color_slope7				
Driver_Setup8	A7h	20h	fader_src8		loglin8	color_slope8				
Driver_Setup9	A8h	20h	fader_src9		loglin9	color_slope9				
Start_Addr1	B0h	00h		start_addr1						
Start_Addr2	B1h	00h		start_addr2						
Start_Addr3	B2h	00h		start_addr3						
Seq1_PC	B4h	00h		PC1						
Seq2_PC	B5h	00h		PC2						
Seq3_PC	B6h	00h		PC3						
Variable_A1	B8h	00h	var_a1							

Table 107. Register Map (Continued)

Register Definition	Addr hex	Default	Content							
Name			**b7**	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**b0**
Variable_A2	B9h	00h	var_a2							
Variable_A3	BAh	00h	var_a3							
Variable_C	BBh	00h	var_c							
Variable_B1	Bch	00h	var_b1							
Variable_B2	Bdh	00h	var_b2							
Variable_B3	BEh	00h	var_b3							
SRAM0	D0h	00h	sram_0							
SRAM1	D1h	00h	sram_1							
SRAM2	D2h	00h	sram_2							
SRAM3	D3h	00h	sram_3							
SRAM4	D4h	00h	sram_4							
SRAM5	D5h	00h	sram_5							
SRAM6	D6h	00h	sram_6							
SRAM7	D7h	00h	sram_7							
SRAM8	D8h	00h	sram_8							
SRAM9	D9h	00h	sram_9							
SRAM10	Dah	00h	sram_10							
SRAM11	Dbh	00h	sram_11							
SRAM12	Dch	00h	sram_12							
SRAM13	Ddh	00h	sram_13							
SRAM14	Deh	00h	sram_14							
SRAM15	Dfh	00h	sram_15							
Program_Direct_Access	FEh	00h	96x16_bits_instruction_code see Program Direct Access on page 43							

Register is R/W

Register is read-only

Table 108. Information Registers (only for demoboard software)

Register Definition	Addr hex	Default	Content							
Name			**b7**	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**b0**
CP_Mode_Switch	06h	00h	LED9_high_volt	LED9_low_volt	see Table 107 on page 68					
LED_Low_Voltage_Status	07h	00h	LED8_low_volt	LED7_low_volt	LED6_low_volt	LED5_low_volt	LED4_low_volt	LED3_low_volt	LED2_low_volt	LED1_low_volt

Table 108. Information Registers (only for demoboard software) (Continued)

Register Definition	Addr hex	Default	Content							
Name			**b7**	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**b0**
Temp_Sense_ Control	0Eh	00h					cp_skip_ status			
Audio_AGC	1Dh	00h	audio_di s_start	audio_m an_start						
LED_High_Voltage _Status	1Eh	00h	LED8_lhi gh_volt	LED7_hi gh_volt	LED6_hi gh_volt	LED5_hi gh_volt	LED4_hi gh_volt	LED3_hi gh_volt	LED2_hi gh_volt	LED1_hi gh_volt
Mux1_LSB	20h	00h	s1_led8	s1_led7	s1_led6	s1_led5	s1_led4	s1_led3	s1_led2	s1_led1
Mux2_LSB	21h	00h	s2_led8	s2_led7	s2_led6	s2_led5	s2_led4	s2_led3	s2_led2	s2_led1
Mux3_LSB	22h	00h	s3_led8	s3_led7	s3_led6	s3_led5	s3_led4	s3_led3	s3_led2	s3_led1
Mux1_MSB	24h	00h	s1_gpo							s1_led9
Mux2_MSB	25h	00h	s2_gpo							s2_led9
Mux3_MSB	26h	00h	s3_gpo							s3_led9
Trigger_Wait1	28h	00h					ext_trigg er	ch3_trigg er	ch2_trigg er	
Trigger_Wait2	29h	00h					ext_trigg er	ch3_trigg er		ch1_trigg er
Trigger_Wait3	2Ah	00h					ext_trigg er		ch2_trigg er	ch1_trigg er
Audio_Result	2Fh	00h	audio_result							
Page_Select	5Fh	00h					loop_cou nter_sele ct	see Table 107 on page 68		
Table1_StartAddr	C4h	00h		table_start1						
Table2_StartAddr	C5h	00h		table_start2						
Table3_StartAddr	C6h	00h		table_start3						
Table1_EndAddr	C8h	00h		table_end1						
Table2_EndAddr	C9h	00h		table_end2						
Table3_EndAddr	Cah	00h		table_end3						
Table1_Pointer	Cch	00h		table_ptr1						
Table2_Pointer	Cdh	00h		table_ptr2						
Table3_Pointer	Ceh	00h		table_ptr3						

Register is R/W

Register is read-only

12 Application Information

External Components

Low ESR input capacitors reduce input switching noise and reduce the peak current drawn from the battery. Low ESR output capacitors should be used to minimize VOUT ripple.

Ceramic capacitors are required and should be located as close to the device as is practical. X5R dielectric material is recommended due to their ability to maintain capacitance over wide voltage and temperature range.

Input, Output and C_{2V5} Capacitor

Table 109. Recommended Input, Output and C_{2V5} Capacitor

Name	Part Number	C	TC Code	Rated Voltage	Size	Manufacturer
C_{BAT}, C_{VCPOUT}, C_{2V5}	GRM188R60J105K	1.0µF +/-15%	X5R	6V3	0603	Murata www.murata.com
	223824613663	1.0µF +/-10%	X5R	10V	0603	Phycomp www.phycomp.com

If a different input capacitor is chosen, ensure similar ESR value and at least 0.6µF capacitance at the maximum input supply voltage. Larger capacitor values (C) for C_{BAT} may be used without limitations.

Flying capacitors

Table 110. Recommended Input, Output and C_{2V5} Capacitor

Name	Part Number	C	TC Code	Rated Voltage	Size	Manufacturer
C_{FLY1}, C_{FLY2}	GRM155R60J474K	470nF +/-15%	X5R	6V3	0402	Murata www.murata.com
	C0603C474K4RAC	470nF +/-10%	X7R	16V	0603	Kemet www.kemet.com

If a different input capacitor is chosen, ensure similar ESR value and at least 0.3µF capacitance at the maximum output voltage. Larger capacitor values (C) may be used without limitations.

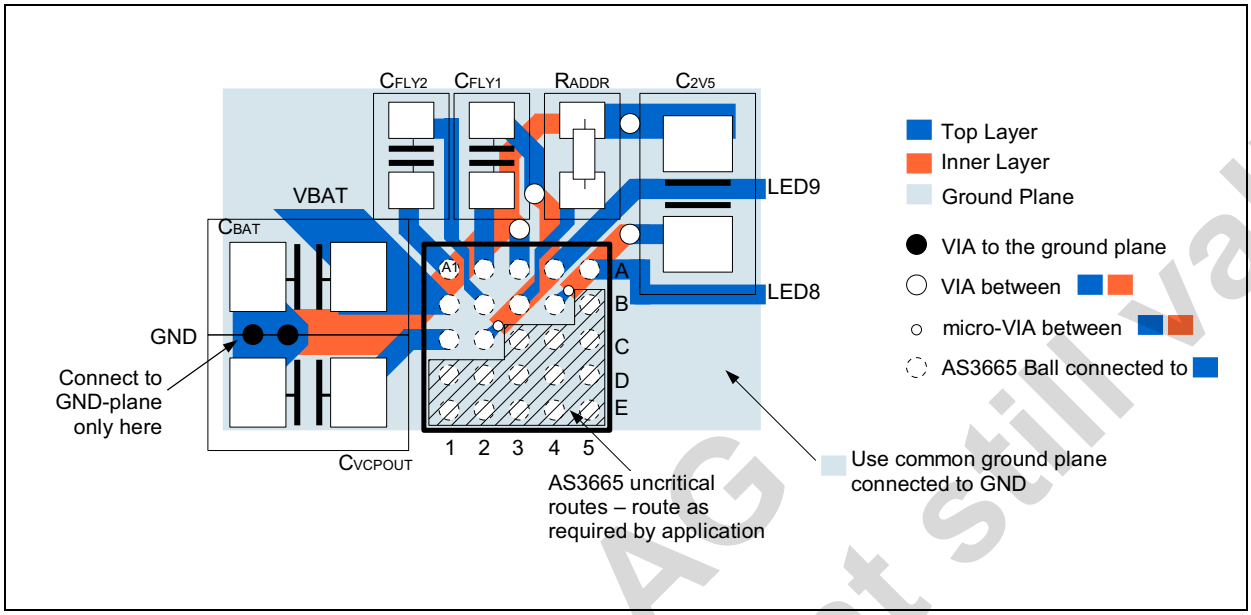
PCB Layout Guideline

The high speed operation requires proper layout for optimum performance. Route the power traces first and try to minimize the area and wire length of the two high frequency/high current loops:

1. C_{BAT} to C_{FLY1} and/or C_{FLY2}
2. C_{FLY1} and/or C_{FLY2} to C_{VCPOUT}

The ground plane of the system should be connected to the layout of the AS3665 only at a single point. This avoid noise to travel from the internal switching node to the application - see Figure 29:

Figure 29. Layout recommendation



Note: If component placement rules allow, move all components close to the AS3665

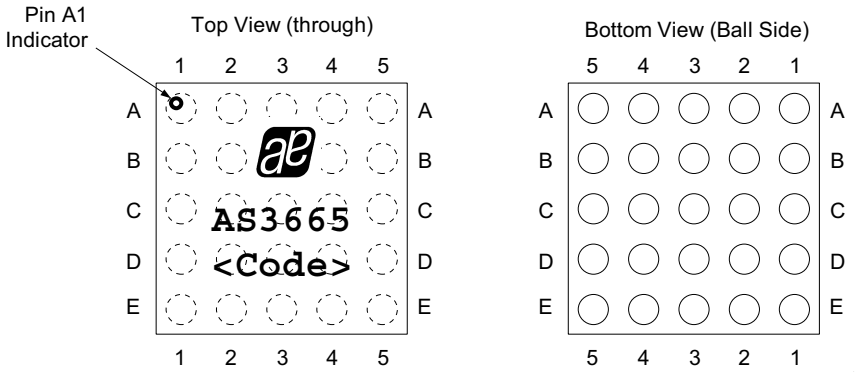
It is possible to route the AS3665 with only two planes to reduce the cost of the PCB.

LED Test

See LED Test on page 39.

13 Package Drawings and Markings

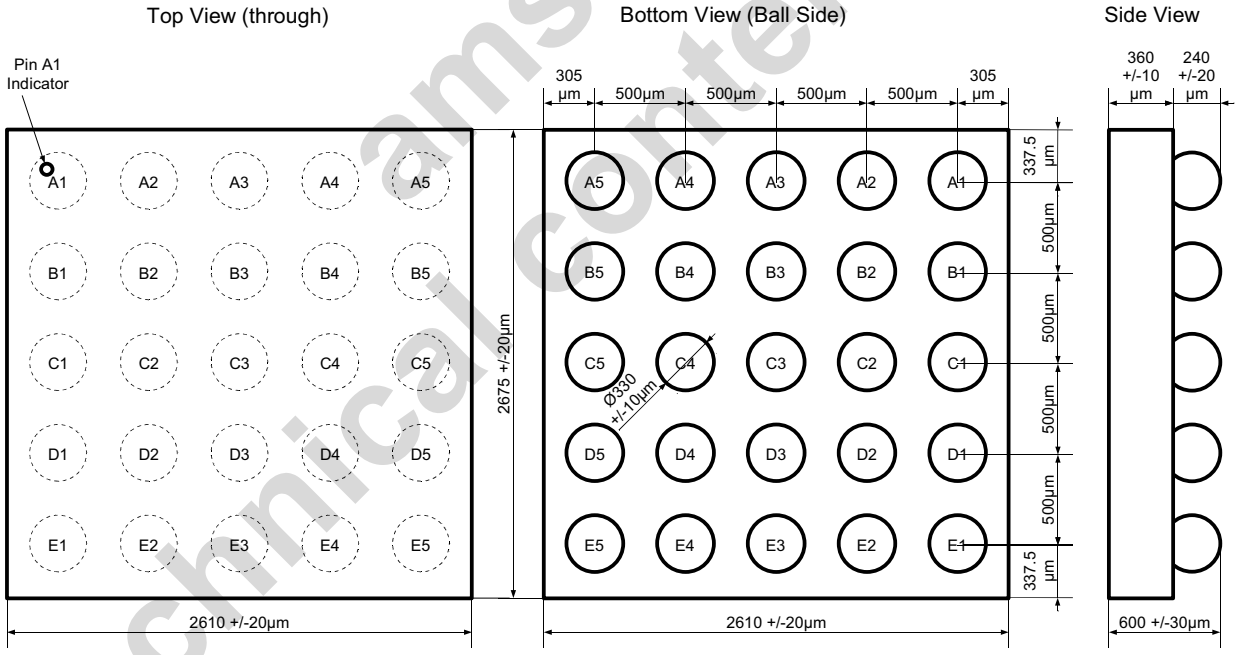
Figure 30. WL-CSP-25 (2.610x2.675mm) 0.5mm pitch Marking



Note:

Line 1: austriamicrosystems logo
Line 2: AS3665
Line 3: <Code>
Encoded Datecode (4 characters)

Figure 31. WL-CSP-25 (2.610x2.675mm) 0.5mm pitch Package Dimensions



The coplanarity of the balls is 40µm.

14 Ordering Information

The devices are available as the standard products shown in Table 111.

Table 111. Ordering Information

Model	Description	Delivery Form	Package
AS3665-ZWLT	9 Channel Advanced Command Driven RGB/White LED Driver	Tape & Reel	WL-CSP-25 (2.610x2.675mm) 0.5mm pitch

Note: AS3665-ZWLT

AS3665-
- Z Temperature Range:
  - Z..... -30ºC - 85ºC
- WL Package Type:
  - WL ..... Wafer Level Chip Scale Package WL-CSP-25 (2.610x2.675mm) 0.5mm pitch
- T Delivery Form:
  - T..... Tape & Reel (no dry pack required)