Application Note

In-Circuit Programming of DA9061/2/3

AN-PM-080

Abstract

The Dialog Semiconductor DA9061/2/3 PMICs can be configured during board-level testing using the Dialog In-Circuit Programmer Kit together with this application note. This provides flexibile device configuration and just-in-time programming.

Contents

Figures

Tables

1 Terms and Definitions

CRC	Cyclic redundancy check (used by Dialog to identify OTP configuration ini files)
DA906x	DA9061, DA9062, DA9063 and DA9063L
DUT	Device under test
ESD	Electrostatic discharge
GUI	Graphical user interface
ICP	In-circuit programming
ini file	Configuration file defining the OTP content
OTP	One-time programmable (memory)
PMIC	Power management IC
seed ini file	The configuration of the partly-configured OTP

2 References

[1] DA9061 Datasheet, Dialog Semiconductor

[2] DA9062 Datasheet, Dialog Semiconductor

[3] DA9063 Datasheet, Dialog Semiconductor

[4] Dialog Support Site, http://www.dialog-semiconductor.com/support

[5] AN-PM-060, DA9061/2 Developers' Guide

[6] UM-PM-008, SmartCanvas DA9061/2 User Manual, Dialog Semiconductor

3 Introduction

The Dialog PMIC architecture provides configuration flexibility so the PMIC can be used in many applications. This flexibility is achieved by incorporating OTP in the DA906x which configures the device at start-up. Programming of the OTP is outlined in Table 1.

Table 1: OTP Programming Options

Programming Method	Suitability	Volume (Units)	Comments
Evaluation kit	Development of PMIC configuration using SmartCanvas™ (Note 1) and Power Commander Mode	1	No need for OTP programming, see SmartCanvas DA9061/2 User Manual [5].
	Prototype builds or bespoke small volume manufacture	≤ 30	Evaluation kit and socket board (Note 2).
In-circuit programming (ICP)	Volume manufacturing	< 30k	
Dialog custom variant	High volume manufacturing	> 30k	Usually supplied directly from Dialog as a custom OTP variant.
Dialog standard variants	Platform-specific	≥ 1	Supplied with the OTP programmed for specific platforms such as NXP i.MX 5/6/7/8 families, Xilinx Zynq™, Renesas R-Car™.

Note 1 SmartCanvas is the GUI software included with the Dialog PMIC Evaluation Kits. The software can be downloaded separately from the Dialog Support Site [4].

Note 2 Socket boards are DA9063-EVAL6, DA9061-SOCKETBOARD, and DA9062-SOCKETBOARD

Dialog manufactures an expanding range of standard variants that support specific platforms such as NXP i.MX 5 and 6 families, Atmel SAMA5, and Renesas R-Car. Dialog recommends the use of these standard variants where available. Other platforms are supported by programming the PMIC OTP as described in Table 1. ICP is the most suitable programming solution for fewer than 30k devices. ICP has the advantage of low additional cost once the programming interface has been designed and avoids manual handling of the devices. With ICP, the PMICs follow the standard board assembly. Board rework due to PMIC programming failure is expected to be insignificant due to the high quality of Dialog products: expected programming failure rate is measured in parts-per-million and expected programming yield is therefore close to 100 %.

3.1 ICP

Dialog supplies seed ini files which simplify the configuration process for system designers. These files allow start-up customization for parameters such as regulator voltage, regulator power-up / power-down sequence, voltage monitoring, and GPIO configuration.

Many other settings are configurable by I^2C writes to the PMIC after the system starts, as described in the datasheets [1], [2], and [3].

The basic premise of the ICP OTP is to add the application-specific settings into the seed ini file, so to create a complete OTP configuration (Figure 1).

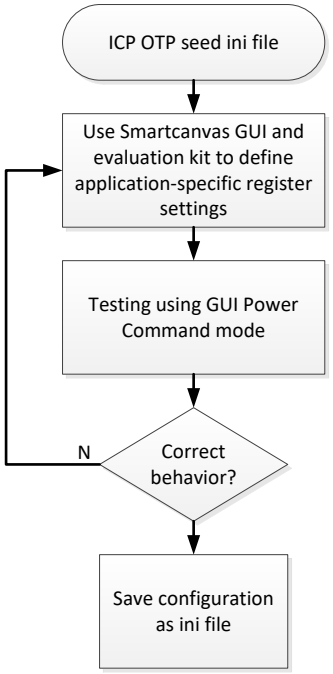

Figure 1: Principle for Defining PMIC Application-Specific Settings

Once the configuration has been finalized and tested, production is set up to repeat the programming of the custom settings into the blank DA906x-00 parts (Figure 2).

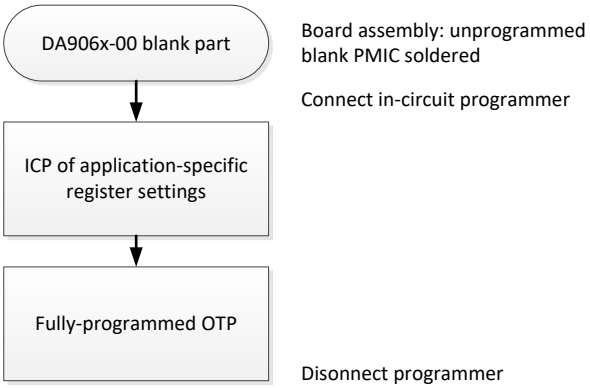

Figure 2: Principle of ICP

4 Development Flow

Expanding on Figure 1, the first step is to customize the OTP configuration for the target application. This is achieved by:

1. Using the SmartCanvas GUI and evaluation kit:
   a. Select a suitable seed ini file, see Table 2 (contact Dialog support for DA9063 seed file availability).
   b. Download the ini file from the Dialog Support Site [4] and open it using SmartCanvas.
   c. Determine the correct settings for the registers discussed in Section 5 and Appendix A.
   d. Finalize the OTP configuration and save it as a new ini file. Make a note of the CRC.
2. Set up and test the programmer hardware interface and test the In-Circuit Programmer
3. Set up production

Table 2: OTP Seed Files

Device	Seed ini File	Description (Note 1)
DA9061	DA9061_seed_for_in-cct_prog_v02-2602.ini	Starting point for a new DA9061 OTP configuration. Basic configuration registers need to be set.
DA9062	DA9062_seed_for_in-cct-prog_v02-09A6.ini	Starting point for a new DA9062 OTP configuration. Basic configuration registers need to be set.
DA9063	DA9063_seed_for_in-cct_prog_v01-8D23.ini	Starting point for a new DA9063 OTP configuration. Basic configuration registers need to be set.
DA9063L	DA9063L_seed_for_in-cct_prog_v01-1784.ini	Starting point for a new DA9063 OTP configuration. Basic configuration registers need to be set.

Note 1 The seed ini files exclude settings for the basic configuration registers, see Section 5.1.

The latest seed ini files are on the Dialog Support Site. The files are also on the USB stick which is included in the In-Circuit Programmer Kit.

5 PMIC Configuration

Dialog PMICs allow customization of many parameters and device behavior. To assist the developer, the sub-sections below separate registers that provide customization of basic configuration, those that are intermediate, and advanced features. The SmartCanvas GUI provides register information when a register control is highlighted with the mouse pointer: this is an efficient quick reference when working on configurations.

5.1 Basic Configuration Registers

The registers are listed in Table 3. For the regulator control, the basic configuration uses only the A regulator settings. The B settings are not used. After setting the voltages, the power-up and power-down sequence needs to be configured. The sequencer registers are listed in Table 3, although it is more convenient to set these values using the SmartCanvas Power Sequencer utility that is found on the front panel of the GUI (Figure 3).

Table 3: DA9062 Basic Configuration Register Settings

Register Address	Register Name	Control Name	Purpose
0x010	CONTROL_C	AUTO_BOOT	Determines if the system immediately boots up through the sequence to the ACTIVE state or if it enters POWERDOWN mode and waits for a wake-up such as an ONKEY press.
0x0A3	VBUCK2_A	VBUCK2_A	Sets the Buck2 voltage
0x0A4	VBUCK1_A	VBUCK1_A	Sets the Buck1 voltage
0x0A5	VBUCK4_A	VBUCK4_A	Sets the Buck4 voltage
0x0A7	VBUCK3_A	VBUCK3_A	Sets the Buck3 voltage
0x0A9	VLDO1_A	VLDO1_A	Sets the LDO1 voltage
0x0AA	VLDO2_A	VLDO2_A	Sets the LDO2 voltage
0x0AB	VLDO3_A	VLDO3_A	Sets the LDO3 voltage
0x0AC	VLDO4_A	VLDO4_A	Sets the LDO4 voltage
0x082	SEQ_TIMER	SEQ_TIME	Sets the sequencer slot period
0x082	SEQ_TIMER	SEQ_DUMMY	Sets the sequencer dummy slot period
0x083	ID_2_1	LDO1_STEP	Sets the LDO1 sequencer slot (Note 1)
0x083	ID_2_1	LDO2_STEP	Sets the LDO2 sequencer slot (Note 1)
0x084	ID_4_3	LDO3_STEP	Sets the LDO3 sequencer slot (Note 1)
0x084	ID_4_3	LDO4_STEP	Sets the LDO4 sequencer slot (Note 1)
0x089	ID_14_13	BUCK1_STEP	Sets the Buck1 sequencer slot (Note 1)
0x089	ID_14_13	BUCK2_STEP	Sets the Buck2 sequencer slot (Note 1)
0x08A	ID_16_15	BUCK4_STEP	Sets the Buck4 sequencer slot (Note 1)
0x08A	ID_16_15	BUCK3_STEP	Sets the Buck3 sequencer slot (Note 1)
0x095	SEQ_A	SYSTEM_END	Recommended value for a basic configuration is 0x1 (Note 2)
0x095	SEQ_A	POWER_END	Recommended value for a basic configuration is 0xE (Note 2)
0x096	SEQ_B	MAX_COUNT	Recommended value for a basic configuration is 0xF (Note 2)
0x096	SEQ_B	PART_DOWN	Recommended value for a basic configuration is 0x0 (Note 2)
0x099	RESET	RESET_TIMER	Configures when the RESET_EVENT timer begins
0x099	RESET	RESET_EVENT	Configures the delay for nRESET being released. The timer start is determined by the setting of RESET_TIMER
0x0C5	BBAT_CONT	BCHG_VSET	Sets the backup battery voltage
0x0C5	BBAT_CONT	BCHG_ISET	Sets the backup battery charging current

Note 1 Setting the control to 0x0 as Slot 0 means the regulator is not sequenced. The default behavior is then that the regulator remains turned off.

Note 2 These pointers must be ordered PART_DOWN < SYSTEM_END < POWER_END < MAX_COUNT

Figure 3: SmartCanvas Power Sequencer

5.2 Intermediate Configuration Registers

The intermediate configuration registers are highlighted in Table 7, Appendix A.

5.3 Advanced Configuration Registers

There are many other registers available for customization. These can be considered advanced features which require a full understanding of the device. Developers should refer to the datasheet Functional Description sections to understand the behavior of the advanced features, see [1], [2], or [3].

5.4 Serialization Feature and General Purpose Registers

Registers CUSTOMER_ID and CONFIG_ID are programmed by Dialog with the OTP variant number. This provides traceability as to which variant was manufactured by Dialog. Register GP_ID_0 holds the Dialog revision number of this specific variant.

GP_ID_1 to GP_ID_3 can be used by the developer to store other configuration information such as the board revision number.

GP_ID_4 to GP_ID_9 are used by the Dialog ICP tool to store serialization data. This can provide the developer and system production engineers with individual board traceability. Values can be set by the production line operative or an initial value can be entered and the ICP tool will then auto-increment the value for each device programmed. The programmed values can be checked against the Dialog In-Circuit Programmer log files.

To enable a fixed CRC to be calculated, the ICP GUI and Command Line Tool both have the option to mask the serial number. This allows a CRC to be calculated as though GP_ID_4 to GP_ID_9 were fixed as 0x00. This should produce the same CRC as the ini file. However, the correct programming of GP_ID_4 to GP_ID_9 is still checked.

GP_ID_10 to GP_ID_19 remain available for any general purpose values. These values persist through the PMIC RESET mode.

See Section 6.2 for further information about the serialization feature.

5.4.1 Record of Last Serial Number

The last serial number used by the GUI is stored on disk to `next_serial_number_to_program.txt`.

6 Programming Interface

6.1 Hardware

The required connections between the application and the Programmer Board are listed in Table 4 and Table 5, and illustrated in Figure 4 and Figure 5. All five lines can be taken from the J11 header using an 8 x 2 female connector.

The VSYS connection is not used on the 232-18-A revision of the Programmer Board, but may be connected and reserved for future use. However, if connecting VSYS to the 232-18-A Programmer Board then the jumper on J10 pins 23-24 must be removed to prevent damage to the Programmer Board.

Table 4: Connections to Programmer Board for DA9061/2

PMIC Pin on Application Board	Connection on Programmer Board (232-18-A)
GND	Header J11, pin 40
TP	Header J11, pin 26
SDA	Header J11, pin 30 (labelled SDA_0 / SI)
SCL	Header J11, pin 32 (labelled SCL_0 / SK)
VSYS	Header J11, pin 25 (labelled GPIO_11) Note 1

Note 1 The VSYS connection is not used on the 232-18-A revision of the Programmer Board, but VSYS may be connected and reserved for future use. If connecting VSYS to the 232-18-A Programmer Board then the jumper on J10 pins 23 to 24 must be removed to prevent damage.

Table 5: Connections to Programmer Board for DA9063

PMIC Pin on Application Board	Connection on Programmer Board (232-18-A)
GND	Header J11, pin 40
TP	Header J11, pin 26
GPIO14	Header J11, pin 31 (labelled SDA_1)
GPIO15	Header J11, pin 33 (labelled SCL_1)
VSYS	Header J11, pin 25 (labelled GPIO_11) Note 1

Note 1 The VSYS connection is not used on the 232-18-A revision of the Programmer Board, but VSYS may be connected and reserved for future use. If connecting VSYS to the 232-18-A Programmer Board then the jumper on J10 pins 23-24 must be removed to prevent damage.

From a hardware perspective, the programming sequence is:

4. Connect the Programmer Board to the application board. This correctly drives TP high before ramping V_{SYS}. This sequencing is critical for entering the correct programming mode. Note that switch SW1 is not used and should be permanently set to 'ON'.
5. The application should then be supplied with power to raise V_{SYS} to its normal operating level.
6. Start programming using the GUI.

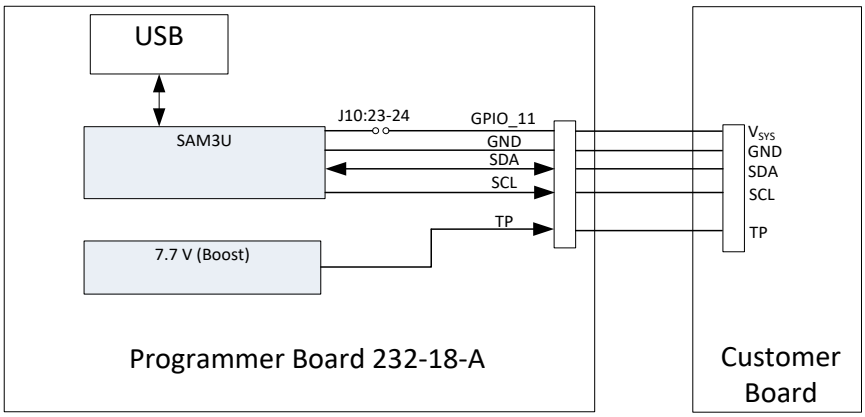

Figure 4: Programmer Board (232-18-A) and Customer Board Ribbon Cable Connection (DA9061/2)

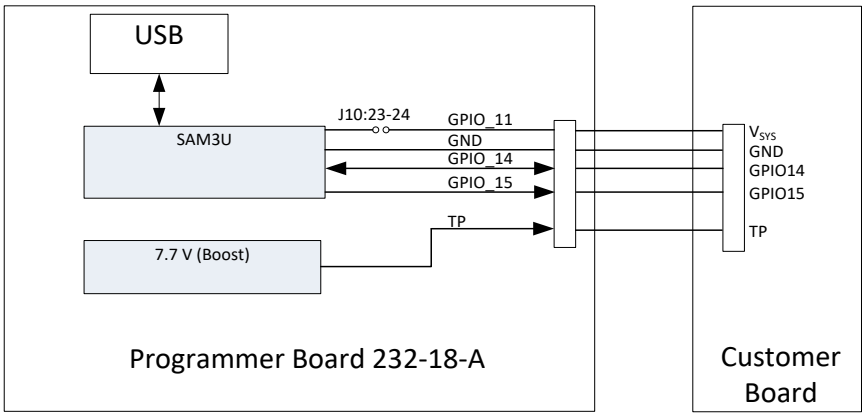

Figure 5: Programmer Board (232-18-A) and Customer Board Ribbon Cable Connection (DA9063)

6.1.1 Application Hardware Design Considerations

When designing the system, the following should be considered:

- V_{SYS}: it should be possible to supply V_{SYS} from the application when required for programming. Since the PMIC does not start up during the programming procedure, it is expected that powering V_{SYS} should be possible without powering the remainder of the system into an unknown state.
  - For production line automation, a voltage detected on the TP pin of $4.5 < V_{TP} < 7.8$ V can be used as a trigger to raise V_{SYS} automatically. A delay of >10 ms is recommended before raising V_{SYS}.
- SCL and SDA: The DA9061/2 application design must allow the I^2C bus to be driven from the Programmer Board. For DA9063, the GPIO14 and GPIO15 lines must be controllable from the Programmer Board. The SCL and SDA lines have 2.2 kΩ pull-ups on the Programmer Board. However, the following must be considered for DA9061/2:
  - There will be other resistors on these lines in the application that are connected to a power rail which is unpowered during PMIC programming. These resistors therefore act as pull-downs on SDA and SCL during programming. The designer should ensure that the V_{IH} that is obtainable on the bus during programming meets the I^2C specification. This means ensuring the application pull-up resistors are of a value ≫2.2 kΩ.
  - The system must be designed so that other unpowered devices connected to the I^2C bus do not clamp the voltage to ground, for example through their V_{DD} ESD protection diodes. A solution is outlined in Figure 6.

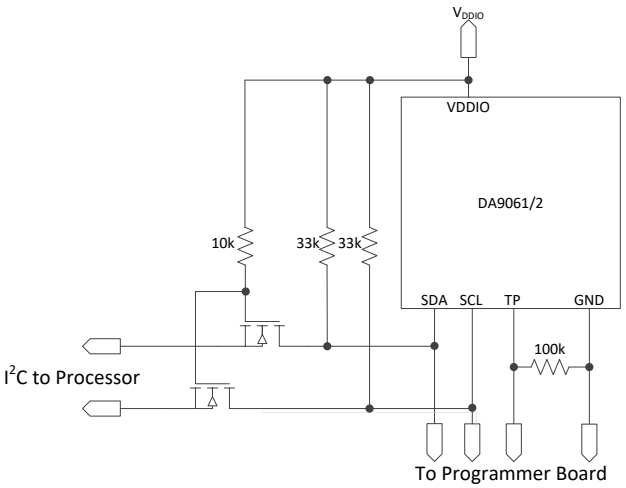

Figure 6: SCL and SDA Isolation in Case of Clamping by the Processor or Other I²C Slaves

- nRESETREQ (DA9061/2), nSHUTDOWN and nOFF (DA9063/L only) and nONKEY: these inputs must remain at a steady state from V_{SYS} being applied until ICP has completed. Any transition on these inputs (including noise sources) may prevent successful programming. Alternatively, they should be driven high by pulling-up to V_{SYS} during programming.

- The cable can be of any design that is most suited to the system board design and production environment – it does not need to be ribbon cable and there is no requirement to retain the 2.54 mm pitch at the application end of the cable: a finer pitch connector may be preferred to save board space.

For system verification, use the specification in Table 6. This is for information only since the Programmer Board is guaranteed to supply the correct programming voltage.

Table 6: Programming Voltage for TP Pin

Parameter	Symbol	Note	Min	Typ	Max	Unit
TP programming voltage	V_{TP_PROG}		7.0	7.5	7.8	V

6.2 Software

Two software programs are used to support the ICP flow. The seed ini files are first customized using SmartCanvas software. After installing the software, a user manual is available from the Help menu.

The In-Circuit Programmer software has been designed for use by production line operatives. The procedure is to:

1. Connect the hardware and start the GUI
2. Open the ini file
3. If required, enable the Serial Number feature and then:
   a. Choose Auto-increment or Manual entry
   b. Set the Current Serial Number value
4. Connect the application board to the ICP Programmer Board. (It is critical to connect the powered In-Circuit Programmer to the powered-off system before powering the system.)
5. The programming sequence is:
   a. Power VSYS supplied from the system. (This must follow after connecting the powered Programmer Board.)

b. Press Scan for Devices: blank devices should be recognized and appear green (Figure 7). A question mark (?) means the DUT is not ready for programming

c. Press the Program OTP button. Programming completion is signified by a green tick (Figure 8)

6. Power down the system board (power down VSYS)
7. Remove the programming interface

ICP can be performed x1, x2, and x4 by connecting additional Programmer Boards to the PC (Figure 9).

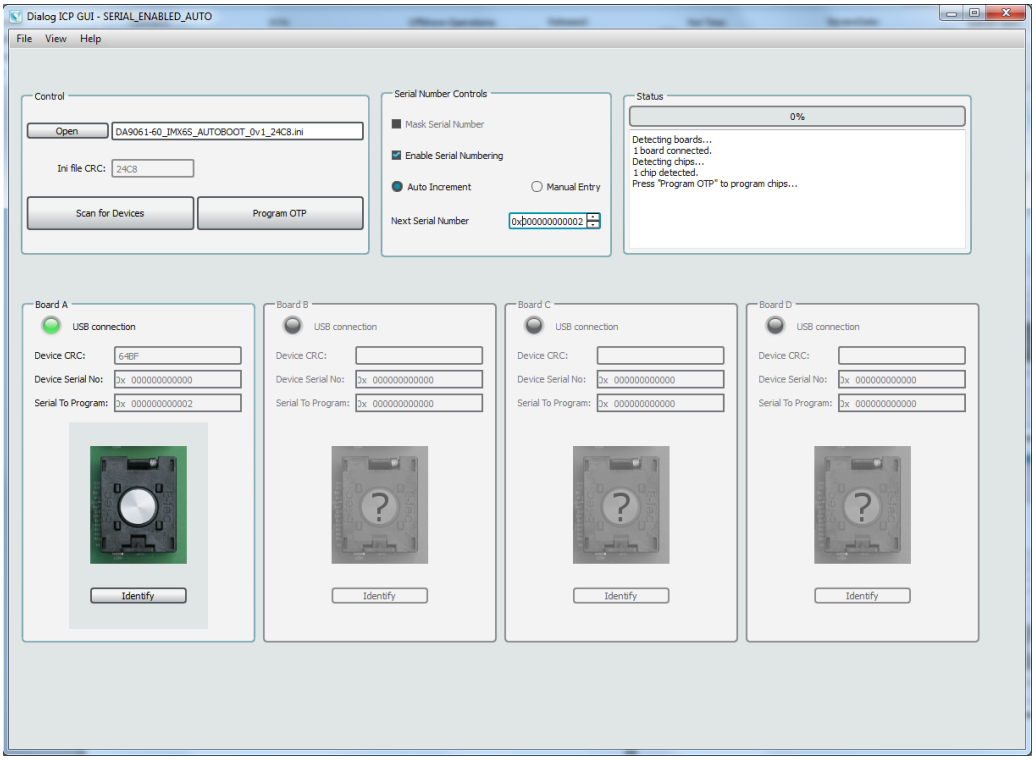

Figure 7: In-Circuit Programmer: Board A Ready to Program

NOTE

If restarting the GUI, the 'Next Serial Number' value will be incremented from the last session (the number is stored in the GUI system file `next_serial_number_to_program.txt` and is loaded when starting the GUI).

If more than a single Programmer Board is connected, the serial numbers for the boards are *n*, *n*+1, *n*+2, *n*+3.

If a device fails to program, its serial number is not re-used.

If using multiple Programmer Boards with manual serial number entry, the serial numbers are entered in the four separate number fields. In this case, the 'Next Serial Number' value is unused.

To enable a fixed CRC to be calculated, the option to 'Mask Serial Number' can be used. This is only necessary when a device has previously been programmed. It is not used when programming a serial number into a blank device. Enabling the 'Mask Serial Number' option allows a CRC to be calculated as though GP_ID_4 to GP_ID_9 were fixed as 0x00. This should produce the same CRC as the ini file.

The 'Mask Serial Number' option is unavailable when 'Enable Serial Numbering' is used. This is because the mask is applied automatically when programming serial numbers.

The auto-increment ends at 0xFFFFFFFFFFFF. It does not wrap to 0x000000000000.

After programming devices with serial numbers, the GUI switches to manual entry and then another refresh is done. The user should wait for this to complete before powering-down the board(s).

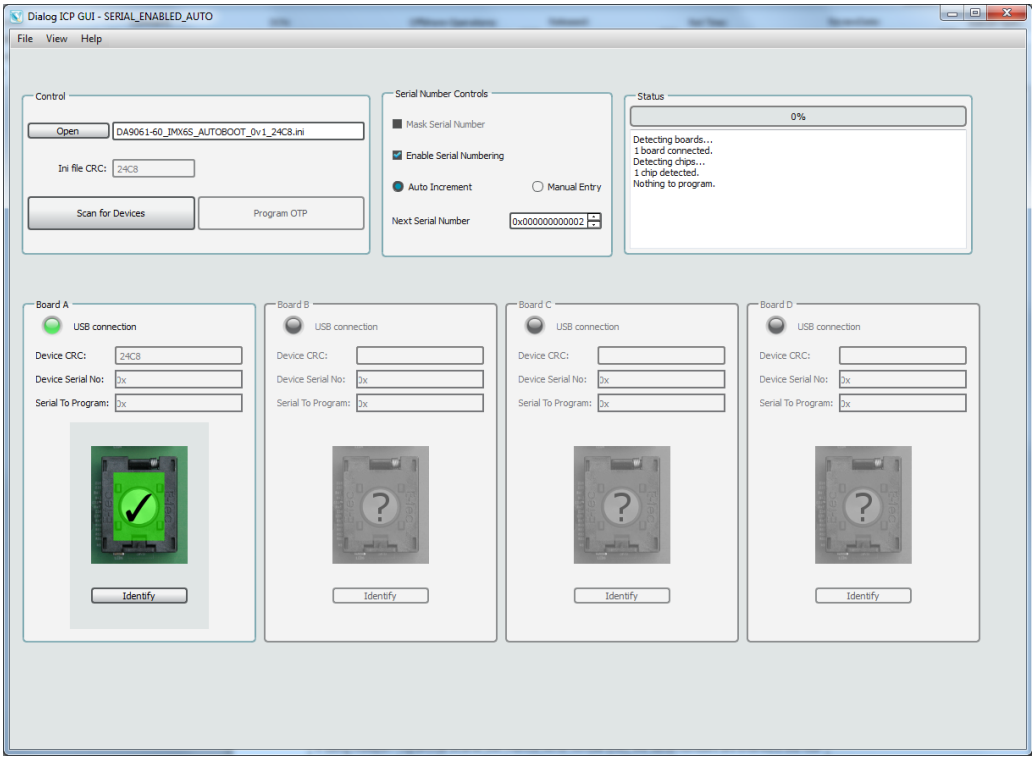

Figure 8: In-Circuit Programmer: Board A Programming Complete

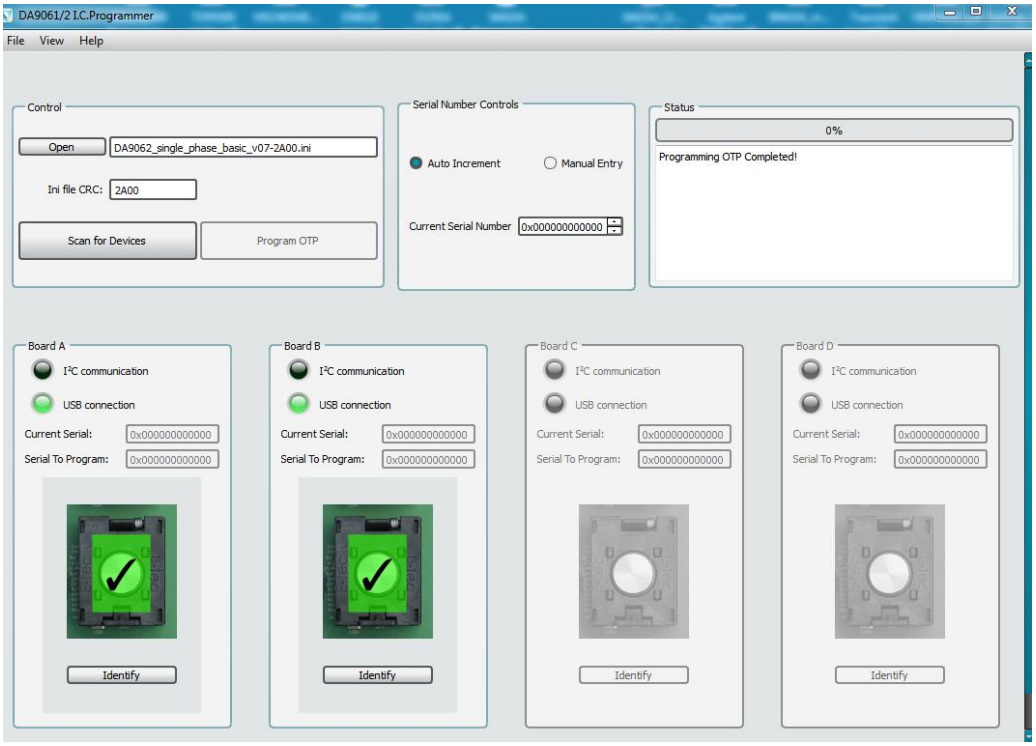

Figure 9: In-Circuit Programmer: Two Boards Connected After Programming Complete

7 Ordering Information

Part Number	Description
DA9061-A0	OTP ready for ICP. Use with customized seed ini files.
DA9062-A0	OTP ready for ICP. Use with customized seed ini files.
DA9063-00	OTP ready for ICP. Use with customized seed ini files.
DA9063L-00	OTP ready for ICP. Use with customized seed ini files.
DA9061_2_3 ICP KIT	ICP kit. Includes board and software.

8 Conclusions

The Dialog DA9061/2/3 PMICs are highly suited to just-in-time programming in the application manufacturing line using the Dialog In-Circuit Programmer Kit.

Appendix A OTP Register List

Table 7 presents the DA9062 registers that require configuration by the system developer. Example configurations are provided by the seed ini files. The table is also valid for DA9061 for all blocks/features that are common with DA9062.

Key:

Reserved

OTP register that contains the basic settings listed in Section 5.1 and intermediate settings that can be customized by the developer

OTP registers that contain advanced settings. These can be customized as described in the datasheet

Table 7: DA9062 Register Summary

Addr	Register	7	6	5	4	3	2	1	0
Page Control									
0x000	PAGE_CON	REVERT	WRITE_MODE	PAGE					
Power Manager Control and Monitoring (except IRQs and events)									
0x001	STATUS_A						DVC_BUSY		NONKEY
0x002	STATUS_B				GPI4	GPI3	GPI2	GPI1	GPI0
0x004	STATUS_D					LDO4_ILIM	LDO3_ILIM	LDO2_ILIM	LDO1_ILIM
0x005	FAULT_LOG	WAIT_SHUT	NRESETREQ	KEY_RESET	TEMP_CRIT	VDD_START	VDD_FAULT	POR	TWD_ERROR
IRQ Events									
0x006	EVENT_A		EVENTS_C	EVENTS_B	E_SEQ_RDY	E_WDG_WARN		E_ALARM	E_NONKEY
0x007	EVENT_B	E_VDD_WARN		E_DVC_RDY		E_LDO_LIM		E_TEMP	
0x008	EVENT_C				E_GPI4	E_GPI3	E_GPI2	E_GPI1	E_GPI0
IRQ Masks									
0x00A	IRQ_MASK_A				M_SEQ_RDY	M_WDG_WARN		M_ALARM	M_NONKEY
0x00B	IRQ_MASK_B	M_VDD_WARN		M_DVC_RDY		M_LDO_LIM		M_TEMP	
0x00C	IRQ_MASK_C				M_GPI4	M_GPI3	M_GPI2	M_GPI1	M_GPI0
System control									
0x00E	CONTROL_A		M_POWER1_EN	M_POWER_EN	M_SYSTEM_EN	STANDBY	POWER1_EN	POWER_EN	SYSTEM_EN
0x00F	CONTROL_B	BUCK_SLOWSTART	NFREEZE		nONKEY_LOCK	NRES_MODE	FREEZE_EN	WATCHDOG_PD	
0x010	CONTROL_C	DEF_SUPPLY	SLEW_RATE		OTPREAD_EN	AUTO_BOOT	DEBOUNCING		

Addr	Register	7	6	5	4	3	2	1	0
0x011	CONTROL_D						TWDSCALE		
0x012	CONTROL_E	V_LOCK					RTC_EN	RTC_MODE_SD	RTC_MODE_PD
0x013	CONTROL_F						WAKE_UP	SHUTDOWN	WATCHDOG
0x014	PD_DIS	PMCONT_DIS	OUT32K_PAUSE	BBAT_DIS	CLDR_PAUSE		PMIF_DIS		GPI_DIS
GPIO control									
0x015	GPIO_0_1	GPIO1_WEN	GPIO1_TYPE	GPIO1_PIN		GPIO0_WEN	GPIO0_TYPE	GPIO0_PIN	
0x016	GPIO_2_3	GPIO3_WEN	GPIO3_TYPE	GPIO3_PIN		GPIO2_WEN	GPIO2_TYPE	GPIO2_PIN	
0x017	GPIO_4					GPIO4_WEN	GPIO4_TYPE	GPIO4_PIN	
0x01C	GPIO_WKUP_MODE				GPIO4_WKUP_MODE	GPIO3_WKUP_MODE	GPIO2_WKUP_MODE	GPIO1_WKUP_MODE	GPIO0_WKUP_MODE
0x01D	GPIO_MODE0_4				GPIO4_MODE	GPIO3_MODE	GPIO2_MODE	GPIO1_MODE	GPIO0_MODE
0x01E	GPIO_OUT0_2	GPIO2_OUT		GPIO1_OUT			GPIO0_OUT		
0x01F	GPIO_OUT3_4				GPIO4_OUT		GPIO3_OUT		
Power supply control									
0x020	BUCK2_CONT		VBUCK2_GPI			BUCK2_CONF	BUCK2_GPI		BUCK2_EN
0x021	BUCK1_CONT		VBUCK1_GPI			BUCK1_CONF	BUCK1_GPI		BUCK1_EN
0x022	BUCK4_CONT		VBUCK4_GPI			BUCK4_CONF	BUCK4_GPI		BUCK4_EN
0x024	BUCK3_CONT		VBUCK3_GPI			BUCK3_CONF	BUCK3_GPI		BUCK3_EN
0x026	LDO1_CONT	LDO1_CONF	VLDO1_GPI			LDO1_PD_DIS	LDO1_GPI		LDO1_EN
0x027	LDO2_CONT	LDO2_CONF	VLDO2_GPI			LDO2_PD_DIS	LDO2_GPI		LDO2_EN
0x028	LDO3_CONT	LDO3_CONF	VLDO3_GPI			LDO3_PD_DIS	LDO3_GPI		LDO3_EN
0x029	LDO4_CONT	LDO4_CONF	VLDO4_GPI			LDO4_PD_DIS	LDO4_GPI		LDO4_EN
0x032	DVC_1	VLDO4_SEL	VLDO3_SEL	VLDO2_SEL	VLDO1_SEL	VBUCK3_SEL	VBUCK4_SEL	VBUCK2_SEL	VBUCK1_SEL
RTC calendar and alarm									
0x040	COUNT_S	RTC_READ		COUNT_SEC					
0x041	COUNT_MI			COUNT_MIN					
0x042	COUNT_H				COUNT_HOUR				
0x043	COUNT_D				COUNT_DAY				
0x044	COUNT_MO					COUNT_MONTH			
0x045	COUNT_Y		MONITOR	COUNT_YEAR					
0x046	ALARM_S	ALARM_STATUS		ALARM_SEC					
0x047	ALARM_MI			ALARM_MIN					
0x048	ALARM_H				ALARM_HOUR				
0x049	ALARM_D				ALARM_DAY				
0x04A	ALARM_MO			TICK_WAKE	TICK_TYPE	ALARM_MONTH			
0x04B	ALARM_Y	TICK_ON	ALARM_ON	ALARM_YEAR					
0x04C	SECOND_A	SECONDS_A							
0x04D	SECOND_B	SECONDS_B							
0x04E	SECOND_C	SECONDS_C							
0x04F	SECOND_D	SECONDS_D							
Power Sequencer									
0x081	SEQ	NXT_SEQ_START				SEQ_POINTER			
0x082	SEQ_TIMER	SEQ_DUMMY				SEQ_TIME			
0x083	ID_2_1	LDO2_STEP				LDO1_STEP			
0x084	ID_4_3	LDO4_STEP				LDO3_STEP			
0x088	ID_12_11	PD_DIS_STEP							
0x089	ID_14_13	BUCK2_STEP				BUCK1_STEP			
0x08A	ID_16_15	BUCK3_STEP				BUCK4_STEP			
0x08D	ID_22_21	GP_FALL1_STEP				GP_RISE1_STEP			
0x08E	ID_24_23	GP_FALL2_STEP				GP_RISE2_STEP			

Addr	Register	7	6	5	4	3	2	1	0
0x08F	ID_26_25	GP_FALL3_STEP				GP_RISE3_STEP			
0x090	ID_28_27	GP_FALL4_STEP				GP_RISE4_STEP			
0x091	ID_30_29	GP_FALL5_STEP				GP_RISE5_STEP			
0x092	ID_32_31	EN32K_STEP				WAIT_STEP			
0x095	SEQ_A	POWER_END				SYSTEM_END			
0x096	SEQ_B	PART_DOWN				MAX_COUNT			
0x097	WAIT	WAIT_DIR		TIME_OUT	WAIT_MODE	WAIT_TIME			
0x098	EN_32K	EN_32KOUT		OUT_CLOCK	DELAY_MODE	CRYSTAL	STABILISATION_TIME		
0x099	RESET	RESET_EVENT		RESET_TIMER					
Power supply control									
0x09A	BUCK_ILIM_A					BUCK3_ILIM			
0x09B	BUCK_ILIM_B					BUCK4_ILIM			
0x09C	BUCK_ILIM_C	BUCK2_ILIM				BUCK1_ILIM			
0x09D	BUCK2_CFG	BUCK2_MODE		BUCK2_PD_DIS					
0x09E	BUCK1_CFG	BUCK1_MODE		BUCK1_PD_DIS					
0x09F	BUCK4_CFG	BUCK4_MODE		BUCK4_PD_DIS	BUCK4_VTT_EN	BUCK4_VTTR_EN			
0x0A0	BUCK3_CFG	BUCK3_MODE		BUCK3_PD_DIS					
0x0A3	VBUCK2_A	BUCK2_SL_A	VBUCK2_A						
0x0A4	VBUCK1_A	BUCK1_SL_A	VBUCK1_A						
0x0A5	VBUCK4_A	BUCK4_SL_A	VBUCK4_A						
0x0A7	VBUCK3_A	BUCK3_SL_A	VBUCK3_A						
0x0A9	VLDO1_A	LDO1_SL_A		VLDO1_A					
0x0AA	VLDO2_A	LDO2_SL_A		VLDO2_A					
0x0AB	VLDO3_A	LDO3_SL_A		VLDO3_A					
0x0AC	VLDO4_A	LDO4_SL_A		VLDO4_A					
0x0B4	VBUCK2_B	BUCK2_SL_B	VBUCK2_B						
0x0B5	VBUCK1_B	BUCK1_SL_B	VBUCK1_B						
0x0B6	VBUCK4_B	BUCK4_SL_B	VBUCK4_B						
0x0B8	VBUCK3_B	BUCK3_SL_B	VBUCK3_B						
0x0BA	VLDO1_B	LDO1_SL_B		VLDO1_B					
0x0BB	VLDO2_B	LDO2_SL_B		VLDO2_B					
0x0BC	VLDO3_B	LDO3_SL_B		VLDO3_B					
0x0BD	VLDO4_B	LDO4_SL_B		VLDO4_B					
BBAT charger control									
0x0C5	BBAT_CONT	BCHG_ISET				BCHG_VSET			
Customer Trim and Configuration									
0x105	INTERFACE	IF_BASE_ADDR							
0x106	CONFIG_A		PM_IF_HSM	PM_IF_FMP	PM_IF_V	IRQ_TYPE	PM_O_TYPE		PM_I_V
0x107	CONFIG_B		VDD_HYST_ADJ			VDD_FAULT_ADJ			
0x108	CONFIG_C		BUCK3_CLK_INV		BUCK4_CLK_INV	BUCK1_CLK_INV	BUCK_ACTV_DIS CHRG		
0x109	CONFIG_D			FORCE_RESET			SYSTEM_EN_RD	NIRQ_MODE	GPI_V
0x10A	CONFIG_E				BUCK3_AUTO		BUCK4_AUTO	BUCK2_AUTO	BUCK1_AUTO
0x10C	CONFIG_G					LDO4_AUTO	LDO3_AUTO	LDO2_AUTO	LDO1_AUTO
0x10D	CONFIG_H		BUCK1_FCM	BUCK2_FCM		BUCK_MERGE			
0x10E	CONFIG_I	LDO_SD	INT_SD_MODE	HOST_SD_MODE	KEY_SD_MODE	WATCHDOG_SD	nONKEY_SD	NONKEY_PIN	
0x10F	CONFIG_J	IF_RESET	TWOWIRE_TO	RESET_DURATION		SHUT_DELAY		KEY_DELAY	
0x110	CONFIG_K				GPIO4_PUPD	GPIO3_PUPD	GPIO2_PUPD	GPIO1_PUPD	GPIO0_PUPD
0x112	CONFIG_M	OSC_FRQ				WDG_MODE		NRESETREQ_PU	
Customer device specific									

Addr	Register	7	6	5	4	3	2	1	0
0x121	GP_ID_0	GP_0 (Note 1)							
0x122	GP_ID_1	GP_1 (Note 1)							
0x123	GP_ID_2	GP_2 (Note 1)							
0x124	GP_ID_3	GP_3 (Note 1)							
0x125	GP_ID_4	GP_4 (Note 2)							
0x126	GP_ID_5	GP_5 (Note 2)							
0x127	GP_ID_6	GP_6 (Note 2)							
0x128	GP_ID_7	GP_7 (Note 2)							
0x129	GP_ID_8	GP_8 (Note 2)							
0x12A	GP_ID_9	GP_9 (Note 2)							
0x12B	GP_ID_10	GP_10							
0x12C	GP_ID_11	GP_11							
0x12D	GP_ID_12	GP_12							
0x12E	GP_ID_13	GP_13							
0x12F	GP_ID_14	GP_14							
0x130	GP_ID_15	GP_15							
0x131	GP_ID_16	GP_16							
0x132	GP_ID_17	GP_17							
0x133	GP_ID_18	GP_18							
0x134	GP_ID_19	GP_19							
0x181	DEVICE_ID	DEV_ID							
0x182	VARIANT_ID	MRC				VRC			
0x183	CUSTOMER_ID	CUST_ID (Note 3)							
0x184	CONFIG_ID	CONFIG_REV (Note 3)							

Note 1 Dialog recommends customers use GP_ID_0 to GP_ID_3 for serialization, ini file configuration traceability, and so on

Note 2 GP_ID_4 to GP_ID_9 are used by the In-Circuit Programmer for production serialization and programming traceability

Note 3 OTP variant number (should be considered read-only). CONFIG_ID must not be 0x00 as the OTP would not be loaded when the PMIC starts. This is not an issue if the value in the seed ini file is not altered.

Appendix B Command Line Tool User Guide

The Command Line Tool provides a method for production line automation. The Command Line Tool replaces the GUI. The Dialog Programmer Board is still required. The tool only works with a single Programmer Board.

B.1 Installation

The Command Line Tool is provided with the In-Circuit Programming Kit on the USB stick. It can also be downloaded from the Dialog Support Site.

The default install folder is `C:\Dialog Semiconductor\Power Management\` Please make a note of your chosen installation folder during the installation process.

The tool cannot be run from the Windows icon. Instead, the tool must be run from a DOS command prompt or called by another program.

B.2 Running the Tool

The tool is called by executing `da906x_icp.exe`. The file extension is optional and therefore the tool can be called using `da906x_icp`. The commands are not case-sensitive. The tool supports three main modes:

PROGRAM: program a device to a specified ini file

VERIFY: verify a device against a specified ini file

DEBUG: debug a device by dumping its OTP contents to a file

Syntax help is available using `da906x_icp -help`

B.2.1 Programming

Syntax: `da906x_icp -program <config.ini>`

Example: `da906x_icp -program DA9061-66_IMX6UL_AUTOBOOT_0v1_1B54.ini`

NOTE
The `-program` option includes verification after programming. The `-verify` option is intended for checking the contents of a device that has previously been programmed.

Calling `da906x_icp.exe` returns one of the codes listed in Table 8.

Table 8: Return Codes

Return Code	Type	Comment
0	PASS	OTP matches .ini file, or .log file written ok
1	USB ERROR	Dialog Programmer Board not detected or driver not installed
2	READ OTP FAILED	No device communication, or OTP read failed
3	INI ERROR	File parse error
4	PROGRAMMING NOT POSSIBLE	OTP already programmed and cannot reprogram
5	PROGRAMMING FAILED	Programming failed and part was unchanged
6	VERIFY FAILED	Non-matching configuration or device partially programmed
7	LOG ERROR	File write/access error
8	PROGRAM ERROR	Exception or other error

B.2.2 Verification

If a device is to be checked without programming, the `-verify` option can be used.

Syntax: `da906x_icp -verify <config.ini>`

Example: `da906x_icp -verify DA9061-66_IMX6UL_AUTOBOOT_0v1_1B54.ini`

B.2.3 Debug

The `-debug` option is used only for debug.

Syntax: `da906x_icp -debug <filename.txt>`

Example: `da906x_icp -debug trace1.txt`

This reads the OTP content and dumps it to the specified filename.

B.2.4 Verbose Mode

The verbose argument is useful for debug.

Examples:

```
da906x_icp -program DA9061-66_IMX6UL_AUTOBOOT_0v1_1B54.ini -verbose
da906x_icp -verify DA9061-66_IMX6UL_AUTOBOOT_0v1_1B54.ini -verbose
```

The information is sent to the standard output, stdout.

B.2.5 Command Line Serialization Feature

As explained in Section 5.4, a number can be stored in the PMIC registers GP_ID_4 to GP_ID_9. This can be used for any customer production purpose. For example, the number can be a static value that represents the system part number or skew. It can be a serial number to provide board-level manufacturing traceability. The value should be generated by the host PC, which might also be the controller for the system tester. The serial number is programmed at the same time as the other OTP data:

Syntax: `da906x_icp -program <config.ini> -serial_num <serial_num>`

Examples:

```
da906x_icp -program DA9061-66_IMX6UL_AUTOBOOT_0v1_1B54.ini -serial_num 002
da906x_icp -program DA9061-66_IMX6UL_AUTOBOOT_0v1_1B54.ini -serial_num AA00FFCC0002
da906x_icp -program DA9061-66_IMX6UL_AUTOBOOT_0v1_1B54.ini -serial_num 0xAA00FFCC0002
```

The number is always parsed as hexadecimal digits. The 0x prefix is optional. Leading zeros are not required.

Once a serial number has been written, bits that remain as zeros can later be overwritten using re-programming (over-programming). It is therefore possible to repeat the programming with a new serial number where bits are changed from 0 to 1. If the new serial number requires bits to be changed from 1 to 0, then the programming will fail and return an error.

B.2.5.1 Masking the Serial Number

It is sometimes necessary to mask the serial number when verifying a device or during re-programming (over-programming). The `-mask_serial_num` option can be used in both situations.

Examples:

```
da906x_icp -program DA9061-66_IMX6UL_AUTOBOOT_0v1_1B54.ini -mask_serial_num
da906x_icp -verify DA9061-66_IMX6UL_AUTOBOOT_0v1_1B54.ini -mask_serial_num
```

The `-serial_num` and `-mask_serial_num` are never used together.

B.2.6 Software Version

The tool version can be recalled:

`Syntax: da906x_icp -version`

This returns the software version, sent to the standard output, stdout.

B.2.7 Command Abbreviations

The commands can be abbreviated.

-h for -help

-p for -program

-v for -verify

-d for -debug

-s for -serial_num

-m for -mask_serial_num

Appendix C SmartCanvas Software Overview

SmartCanvas is included in the Dialog DA906x evaluation kits and is the GUI used with the kit evaluation board. The GUI provides a visual representation of the device registers and of the PMIC operation state. The software (Figure 10) allows interaction with the PMIC, and can be used to read or write to device registers, for testing prototype configurations, and to program socketed devices. The SmartCanvas user manual is accessible via the GUI Help menu and is also available from the Dialog support site [4]. The SmartCanvas software can be separately downloaded from the Dialog Support Site.

Power Commander mode (PC Mode) is a powerful feature of Dialog PMICs that allows prototyping of OTP configurations without having to burn a device's OTP. PC Mode starts a device using the configuration taken from an external ini file instead of the device's own programmed (or blank) configuration. These ini files are compatible with the In-Circuit Programmer and therefore the same ini files can be used to program parts for volume manufacturing.

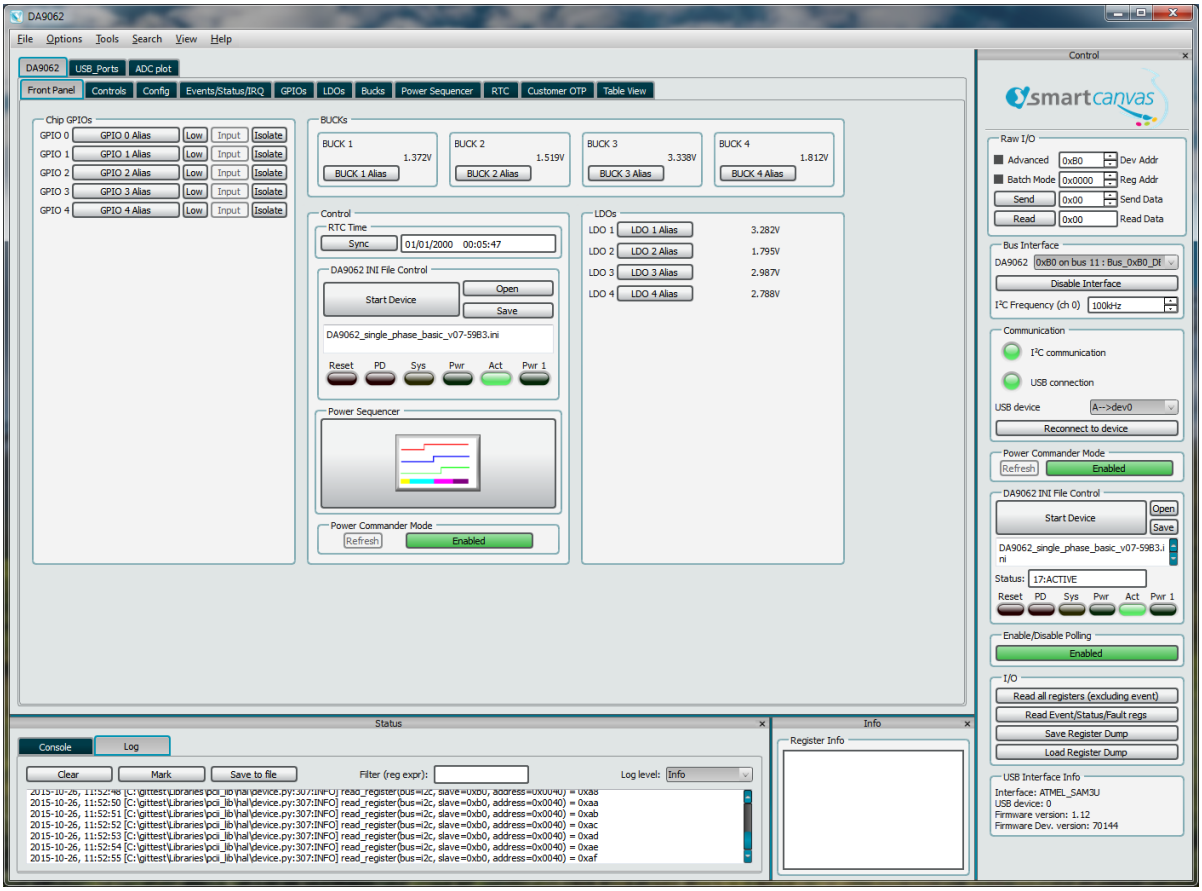

Figure 10: SmartCanvas Front Panel

Revision History

Revision	Date	Description
3.0	18-Feb-2022	• File was rebranded with new logo, copyright and disclaimer
2.0	06-Jun-2018	Section 4: • Added seed ini files for DA9063 and DA9063L Section 7: • Replaced DA9061-00 with DA9061-A0 • Replaced DA9062-00 with DA9062-A0
1.2	15-Mar-2018	• Removed command option abbreviations -verb, -he and -hel
1.1	13-Feb-2018	• Added Serialization feature details to Section 5.4 • Added Command Line Tool description as Appendix B
1.0	21-Aug-2018	• Revised to reflect use of DA906x-00 parts • Revised ini file names in Table 2 • Added Figure 6

Status Definitions

Status	Definition
DRAFT	The content of this document is under review and subject to formal approval, which may result in modifications or additions.
APPROVED or unmarked	The content of this document has been approved for publication.

RoHS Compliance

Dialog Semiconductor's suppliers certify that its products are in compliance with the requirements of Directive 2011/65/EU of the European Parliament on the restriction of the use of certain hazardous substances in electrical and electronic equipment. RoHS certificates from our suppliers are available on request.

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD-PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers who are designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only to develop an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third-party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising from your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.01)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit www.renesas.com/contact-us/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.