Using SPI to Configure and Control the ispPAC®30

January 2002 Application Note AN6027

I. Overview

This application note provides the required documentation to design a system to control and configure the ispPAC30 device via Serial Peripheral Interface (SPI). A summary of the SPI signals is provided, as they relate to the ispPAC30, and the data path from the ispPAC30 SPI shift register to the various internal registers is described. Top-level programming routines are outlined for erase, write, read, and/or verify. Flow charts and timing diagrams are provided at a detailed level for each of the key subroutines. After discussing the programming methods and the SPI timing diagrams, a detailed description of the 112 configuration bits is provided. Bit functions and valid combinations are tabulated, and examples are provided to illustrate how SPI can be used to adjust gain, route input signals, set MDACs, configure output amps (OA)s, and more.

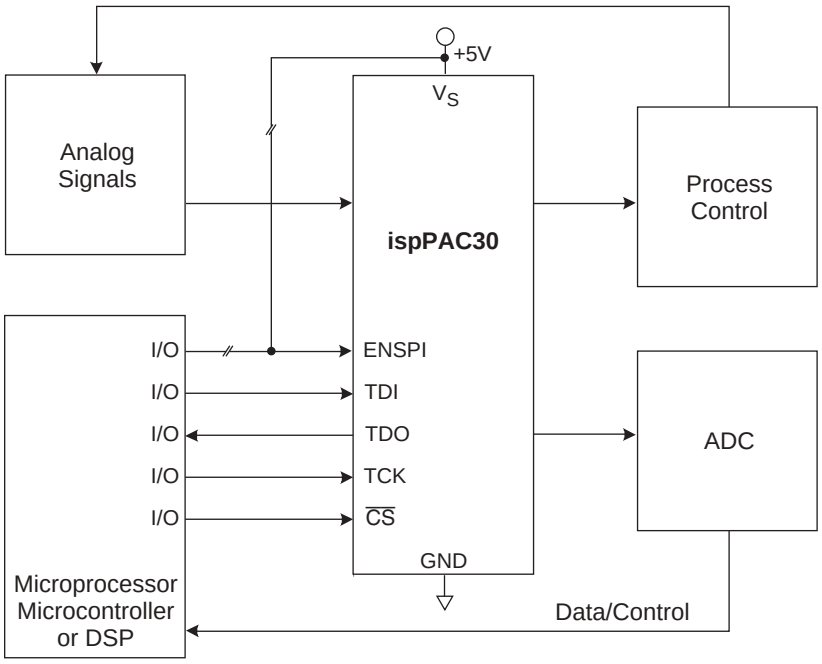
This application note is organized by topics according to the following outline:

2. Introduction

2.1. Serial Peripheral Interface (SPI)

SPI provides a powerful method of controlling virtually every aspect of the ispPAC30 device with as few as four or five wires. The SPI interface hardware inside the ispPAC30 is derived from the JTAG interface, but is optimized to reduce the overhead required to reconfigure the device. Figure 1 illustrates an example application where the ispPAC30 is configured by the I/O ports of an embedded system. One output of the ispPAC30 is used to control a process while the other output is connected to the analog input of an ADC. Note the ENSPI pin can be dynamically controlled from an I/O pin, or statically tied to +5V.

Figure 1. Example SPI Interface to a Microprocessor, Microcontroller or DSP



Configuring the ispPAC30 using the SPI interface consists of controlling the contents within the shift registers using a combination of instructions and data bits. There are five signals involved in the SPI interface. The pins are standard digital I/O pins that can be driven from either a 3.3V or 5V signal. The pins are described in Table 1.

Table 1. Pin Descriptions, SPI Interface

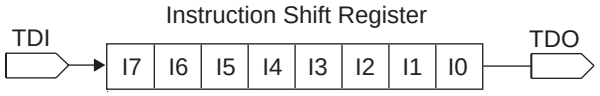
Pin	Description
ENSPI	Enable SPI mode. When ENSPI is HIGH (logic 1), the device is in SPI mode and the SPI instructions will affect the functionality of the device. When ENSPI is LOW (logic 0), the device is in JTAG mode. On the evaluation board and during programming with the Lattice ispDOWNLOAD® cable from PAC-Designer®, the device is in JTAG mode. The evaluation board has a jumper that allows this pin to be pulled to ground for JTAG.
$\overline{CS}$	Chip Select pin. This pin is used to start or end an SPI function, the pin is brought LOW (logic 0) when shifting in data or instructions.
TDI	Standard input pin for shifting data or instructions into the device.
TDO	Data out pin. The data is shifted out TDO on the falling edge of TCK. TDO is tristated when $\overline{CS}$ is HIGH.
TCK	Clock input pin. The rising edge of TCK clocks data into the shift registers. Falling edge clocks data out of shift registers.

These signals can be controlled from any microprocessor, microcontroller, or DSP. The pins can also be driven from a test fixture, or production tester. This allows the board or system to be reconfigured or trimmed during development, prototype, production test or even in the field.

2.2. ispPAC30 Registers

The ispPAC30 is a fully reconfigurable analog circuit. Parameters such as routing, gain, phase, offset, DAC settings and reference voltages can be programmed in a non-volatile mode as well as in an SRAM mode. The device contains a set of registers that hold the configuration bits for the internal architecture. The device functionality is dependent upon the contents of these registers. The E^2CMOS® bits hold a non-volatile copy of the configuration while the device is not powered. When the device is powered up, the configuration bits are automatically copied from E^2 CMOS to SRAM. The outputs of the SRAM registers directly control the various circuits within the ispPAC30. The status or values of the registers can be changed on-the-fly by shifting in the proper instruction and data sequence, thereby achieving unlimited read/write capabilities for re-configuration. The ispPAC30 can decode 32 instructions. However, only 17 instructions are useful when configuring the device using SPI. All the instructions are eight bits long; therefore the instruction register is also eight bits wide, as shown in Figure 2.

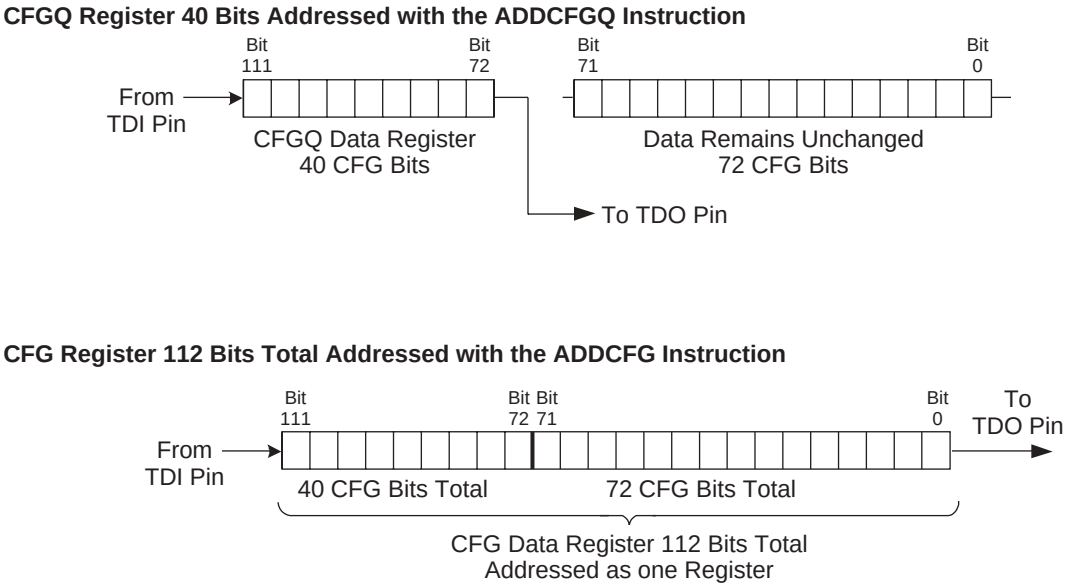
Figure 2. Instruction Shift Register



The 8-bit instruction is shifted into the TDI pin LSB (Least Significant Bit) first, using the rising edge of the clock pin (TCK). The 8-bit word is easy to shift in using an SPI port on a microcontroller or other embedded processor/test system. While the last two bits are being shifted in, the internal state machine decodes and executes the instruction. See Appendix A for a complete list of ispPAC30 instructions.

Figure 3 shows the organization of the configuration (CFG) shift register that is used when reading or writing to the device. The CFG register is addressable in both a short and a long mode. In the quick mode, the upper 40 bits are used to control or read commonly used features such as gain or DAC settings. The 40 bits are shifted into or out of the CFGQ section of the CFG register. In the long mode all 112 CFG bits (40+72) are shifted in and out. This mode allows configuration of “all” the internal architecture.

Figure 3. Configuration Registers



When writing to the CFG register using the ADDCFG instruction, the whole CFG register is filled and each of the 112 bits should be set or cleared, based on the desired functionality of the device. A word of caution when writing in the short mode; the low order bits (0 to 71) need to be set by either shifting in the full 112 bits of data, or reading them from E^2CMOS using the READCFG instruction. Otherwise, they will have an undefined state and the circuit behavior will be unpredictable. For additional documentation regarding the use of the CFGQ, please refer to Appendix C.

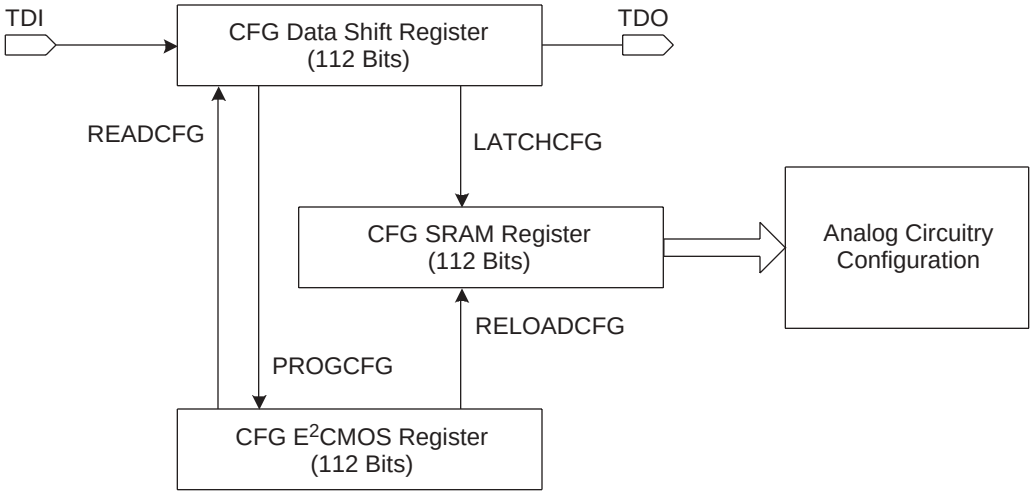
In SPI mode, the TDI and TDO pins can directly connect to several internal registers of the ispPAC30. The primary register is the instruction register, which is connected by bringing the chip select (CS) low. The other registers are connected after decoding the respective instructions. The ADDCFG instruction connects the configuration data shift register; the ADDUES instruction connects the user electronic signature (UES) data shift register; and the IDCODE instruction connects the ID code data shift register. Figure 4 highlights the possible CFG data flow for the internal memory and registers. The arrows show the path that data may take depending on the respective instruction.

CFG Data Shift Register: This is the register where data is shifted in from the TDI pin, and out to the TDO pin. When data is shifted in, it does not affect the device functionality until it is moved into the SRAM configuration register. The pins TDI and TDO both have access to this register after the ADDCFG instruction is decoded. This register transfers data to and from the SRAM and E^2CMOS registers with the appropriate instructions.

CFG SRAM Register: The SRAM register consists of 112 bits of data. It can be written to, from the data shift register using the LATCHCFG instruction or it can be updated from the values stored in E^2CMOS memory, using the RELOADCFG instruction. Either of these instructions will change the value of the SRAM configuration register, which configures the analog circuitry and parameters.

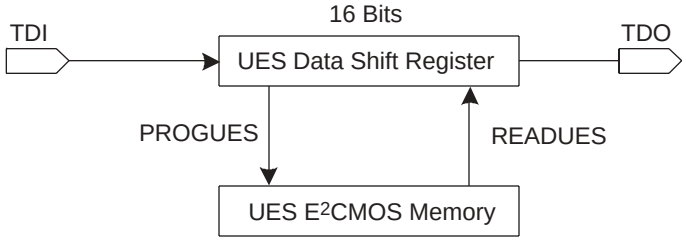
CFG E^2CMOS Register: This non-volatile register also consists of 112 bits of data. It can be written to with the PROGCFG instruction, or read from with the READCFG instruction, and it can transfer its contents to SRAM with the RELOADCFG instruction. At power-up the RELOADCFG instruction is mimicked with internal hardware; thus the analog circuitry can power-up to a known safe state without requiring external intervention.

Figure 4. Data Flow Diagram After Sending ADDCFG Instruction



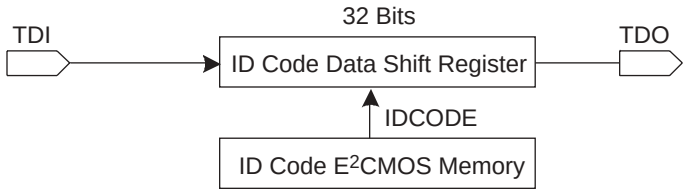
Let us next turn our attention to the UES register. The UES register is a 16-bit portion of the E^2CMOS memory that is used to store device or board information such as set points, date codes or other non-volatile data. The UES register has two instructions for reading or programming. The ADDUES (Address UES) instruction is used to access the register bits. The PROGUES (Program UES) instruction is used to write a value into the E^2CMOS memory. The proper erase instructions and shift timing must be observed in order to program the UES. Figure 5 shows the data flow to and from the UES registers.

Figure 5. Data Flow After Sending ADDUES Instruction



The other register used during programming or re-configuration, is the ID code register. The ID code is a 32-bit string that identifies the device. The ID contains information about the device manufacturer, revision and device type. The data flow for the ID code register is shown in Figure 5. The IDCODE instruction loads the internal 32-bit ID code into the ID code data shift register. The 32 bits of the ID code conform to the IEEE 1149.1 (JTAG) specification. Note that the ID code is read-only and cannot be modified by the user.

Figure 6. Data Flow After Sending IDCODE Instruction



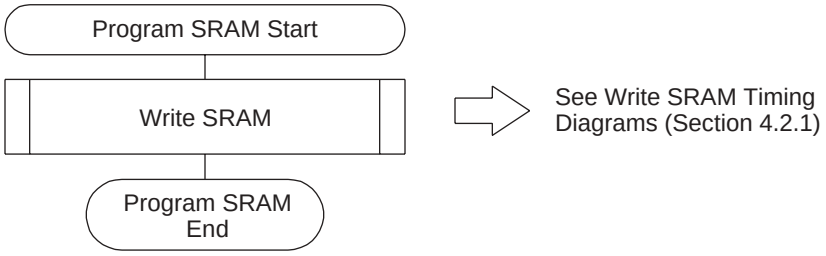
3. SPI Programming Routines

The programming of the ispPAC30 consists of a set of routines that can be sequenced to read, write, verify, or transfer the configuration bits of the various registers. In this section our primary focus will be on the programming of the same registers we looked at in the previous section: the SRAM, E^2CMOS, UES, and ID registers.

3.1. Programming SRAM

Because the analog circuitry is configured and controlled by the contents of the SRAM, the routines to program it will probably be used the most and thus we will start there with Figure 7. From the chart, it can be seen that the contents of the SRAM are not read. This is because they can't be read. It is up to the program, and thus the programmer, to keep an image of the bits in system memory for subsequent modifies and writes.

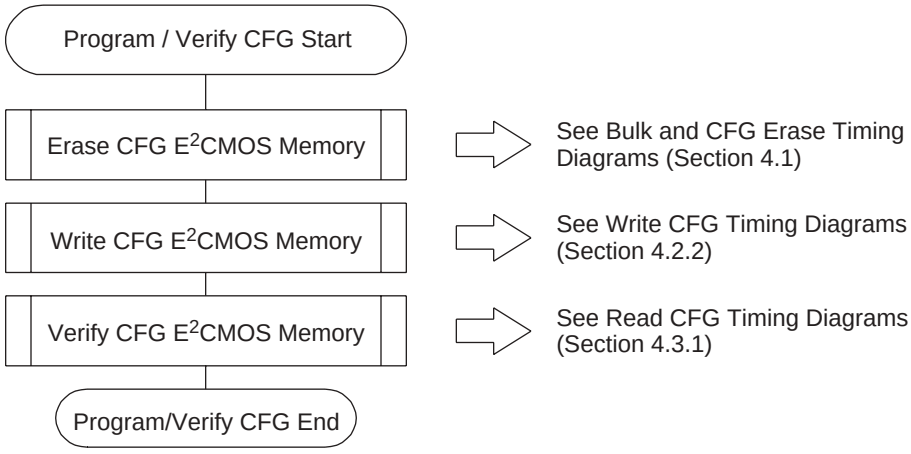
Figure 7. SRAM Programming Flow Chart



3.2. Programming E^2CMOS

Before any changes can be made to the E^2CMOS, it has to be erased, (hence its name electrically erasable) as shown in Figure 8. Initially the E^2CMOS has to have the default or power-up configuration written and verified, so the device will wake up in a circuit-friendly manner. Subsequent rewrites to E^2CMOS may be mandated by systematic or environmental changes, and a verify after a write is always a good idea.

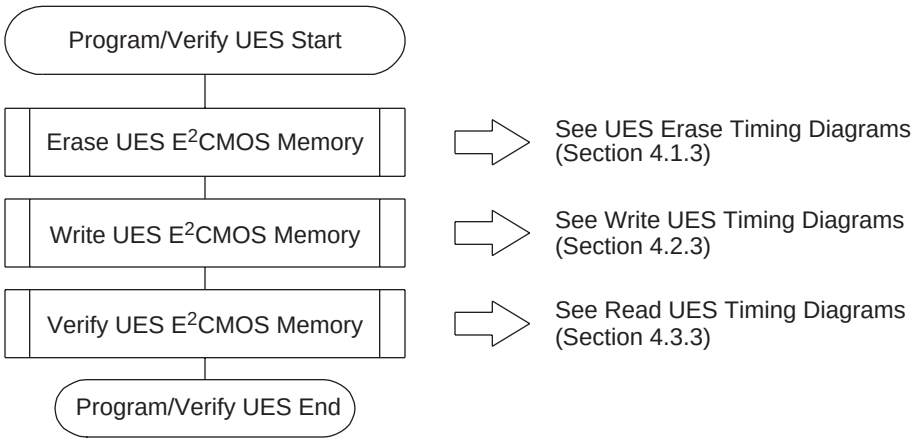
Figure 8. CFG Programming Flow Chart



3.3. Programming the UES

The 16 UES bits are stored in a portion of the E^2CMOS memory, and as was the case in the previous routine, it must first be erased before it can be modified, as shown in Figure 9.

Figure 9. UES Programming Flow Chart



The UES bits are provided as a scratchpad register where the user may store information such as board or assembly ID, production process or test/calibration step. Nevertheless, the state of the UES bits has no effect on the operation of the analog circuitry. Table 2 lists the E^2CMOS bit for each of the UES bits.

Table 2. E^2 Cell Bit List for the UES Register

UES Bit	Bit #
UES_0	0
UES_1	1
UES_2	2
UES_3	3
UES_4	4
UES_5	5
UES_6	6
UES_7	7
UES_8	8
UES_9	9
UES_10	10
UES_11	11
UES_12	12
UES_13	13
UES_14	14
UES_15	15

3.4. Power Down

Two built-in features that can be controlled by single SPI instructions are power consumption and calibration. The ispPAC30 has two global SPI instructions for power control: POWERUP and POWERDN. These instructions switch the ispPAC30 into normal and LOW power standby modes. When the device is powered down using the POWERDN instruction, only the analog circuitry will be powered down; the operating current will typically be 10μA. The ispPAC30 also has an external pin that can be used to power the device down. This pin PD is active LOW. Individual output amplifiers (OA's) in the ispPAC30 can also be controlled for power-down by setting the respective CFG bits with PAC-Designer or SPI. Within the schematic screen of PAC-Designer software, access to the power-down control is located in the menu for changing the OA configuration. For information regarding the power down CFG bits, see section V-H OA Configuration and Appendix B Configuration Bit Table. Refer to the ispPAC30 data sheet for further information on power consumption parameters.

3.5. Auto-Calibration

The ispPAC30 also has a single SPI instruction to initiate the internal calibration sequence. At power on, the ispPAC30 initiates an auto-calibration, which nulls the offsets and has duration of 250 milliseconds. Subsequent re-calibration cycles can be initiated by an external pin CAL that is active HIGH, or by sending the SPI instruction ENCAL. Warm calibration cycles take about 100 milliseconds, during which time the outputs are driven LOW. The common mode voltage that is applied to the various input sections (during calibration) is selectable from two references: GND and +2.5V; thus providing optimal and flexible offset adjustment. For further information regarding calibration see section V-J Auto-Calibration Voltage Settings and the ispPAC30 data sheet.

4. Routines

In this section we develop flow charts and timing diagrams for the various subroutines that are used in the procedures outlined in the previous section. These subroutines can be called in sequences other than those outlined in the previous section; the actual coding is left to the designer and will be dependent upon the system requirements. One caveat that was not known in time for the preliminary data sheet is the required one millisecond delay after the completion of an erase or programming cycle of the E^2CMOS before issuing a read instruction to the same memory. In all the timing diagrams that follow, the ENSPI signal is shown both toggling and staying high to illustrate that the pin can either be controlled by an I/O line or tied high. This section is organized into five parts: erase routines, write routines, read routines, power down routines, and auto-calibration routines.

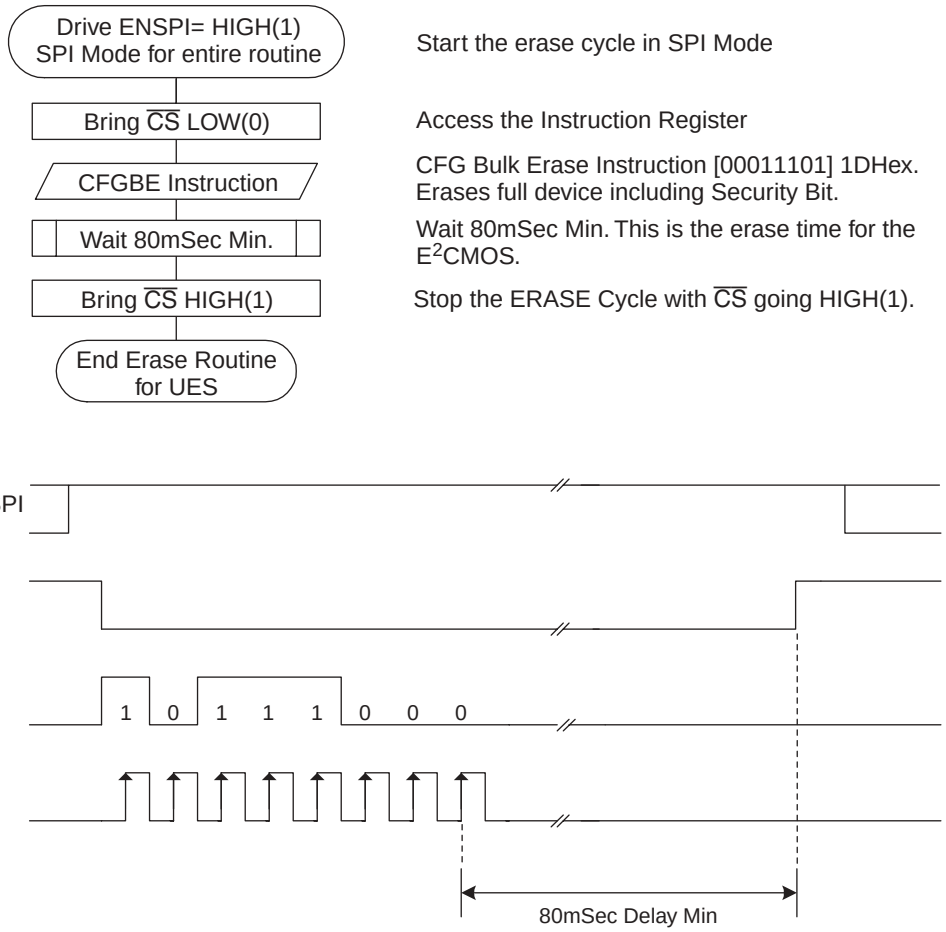
4.1. Erase Routines

The electrically erasable CMOS memory needs to be erased before writing new contents. The subroutines in this section illustrate the usage of instructions that activate the ispPAC30 internal erase circuitry. In each of the following cases, the system must wait a minimum of 80mSec after the last clock for the erase instruction. It is during this time that the E^2CMOS memory is being erased, and $\overline{CS}$ should be held LOW (0) and ENSPI held HIGH (1). Raising $\overline{CS}$ prematurely will interrupt the erase cycle, possibly resulting in an incomplete erasure. When reading after an erasure to verify, $\overline{CS}$ should remain HIGH for one millisecond before sending the read instruction. Note that when writing to the SRAM, erase cycles are not required.

4.1.1 Erase Bulk

This subroutine highlights the usage of the CFGBE instruction that erases the 112 configuration bits, the 16 UES bits, and the electronic security fuse (ESF) bit, as shown in Figure 10. This subroutine would be called before writing to the 112 configuration bits and/or the 16 UES bits.

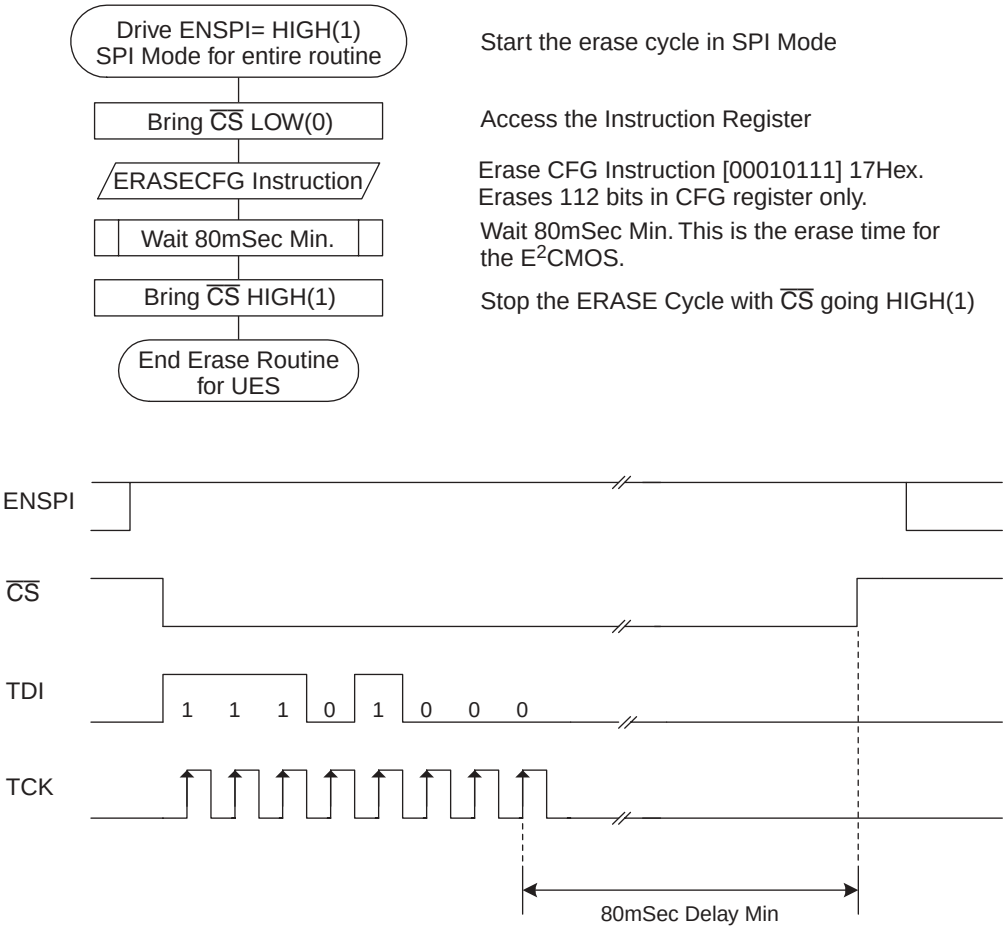
Figure 10. Bulk Erase Flow Chart and Timing Diagram



4.1.2. Erase CFG

This subroutine illustrates the usage of the ERASECFG instruction in Figure 11, which only erases the 112 configuration bits in E^2CMOS. Like the previous subroutine, this would be called before writing any changes to the 112 configuration bits. Note: the UES bits in E^2CMOS memory are not affected by this operation.

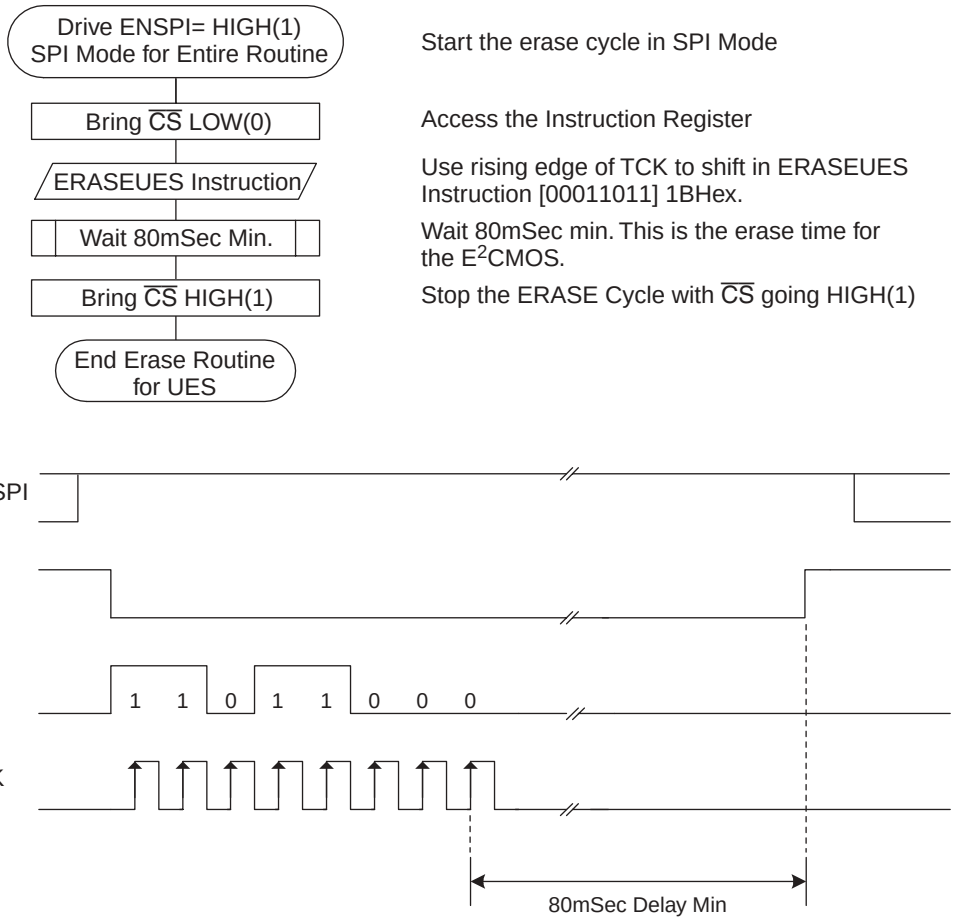
Figure 11. CFG Erase Flow Chart and Timing Diagram



4.1.3. Erase UES

This subroutine focuses on the ERASEUES instruction in Figure 12, which only erases the 16 UES bits of the E^2CMOS, so that a subsequent write of the UES bits may occur. Note: the CFG bits in E^2CMOS memory are not affected by this operation.

Figure 12. UES Erase Flow Chart and Timing Diagram



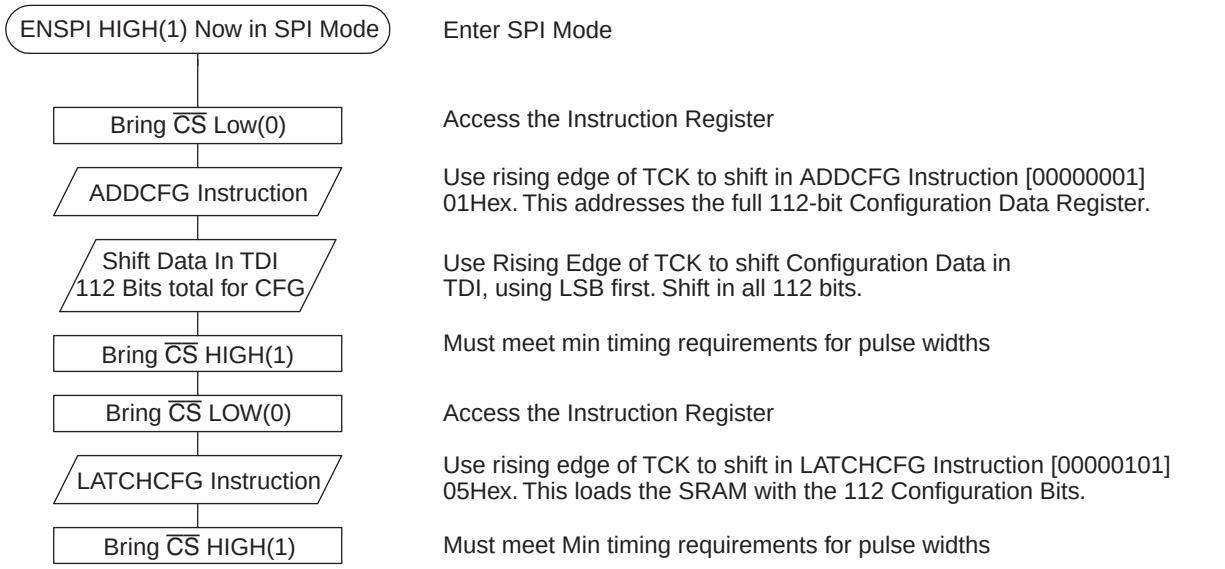
4.2. Write Routines

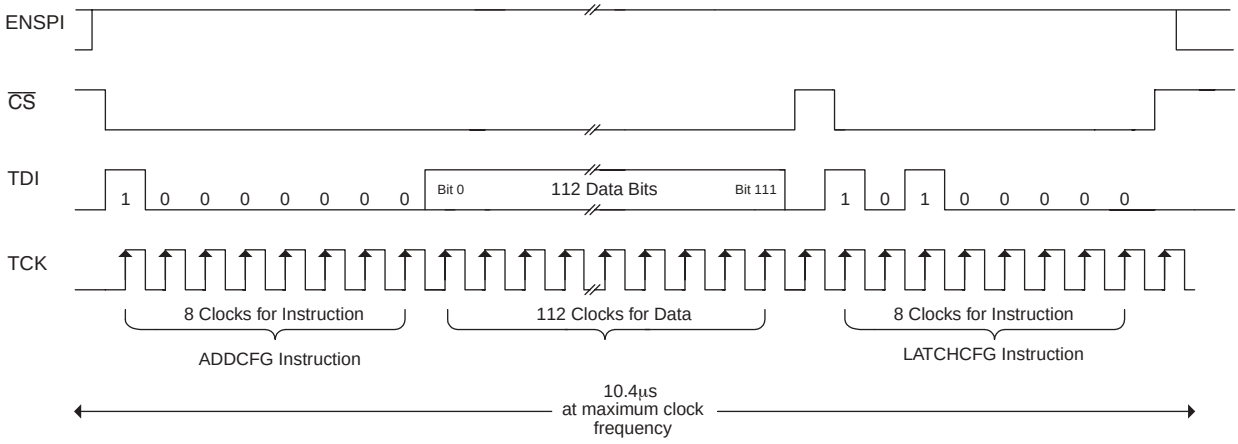
The writing subroutines are used to modify either the configuration bits in SRAM or E^{2}CMOS or the UES bits in E^{2}CMOS. When writing to the SRAM memory, no erase procedure is required. However, when writing to E^{2}CMOS to modify the CFG bits or the UES bits, an erase procedure of the appropriate type is required, preceding the programming instruction. During the program cycle, $\overline{CS}$ must remain LOW for a minimum of 80 milliseconds after the program instruction. Furthermore, when reading after a write is performed to verify, $\overline{CS}$ must remain HIGH for a minimum of one millisecond.

4.2.1. Write SRAM

Once the E^{2}CMOS is configured, it is easy to change the function of the ispPAC30 by loading new CFG data into the SRAM. The basic flow, as illustrated in Figure 13, consists of selecting the device, shifting in the ADDCFG Instruction, followed by the 112 bits of configuration data. The configuration data is then latched into the SRAM memory using the LATCHCFG Instruction. To make another change, place the device in SPI mode again and run the routine with new values. When the write to SRAM is complete, the analog circuitry of the ispPAC30 will conform to the new bit settings. The transfer to SRAM actually occurs during the seventh clock cycle of the LATCHCFG instruction and outputs will settle some time after that. The actual settling time will be dependent upon the configuration of the analog circuitry with the configuration of the output amplifier having the most influence.

Figure 13. SRAM Write Flow Chart and Timing Diagrams

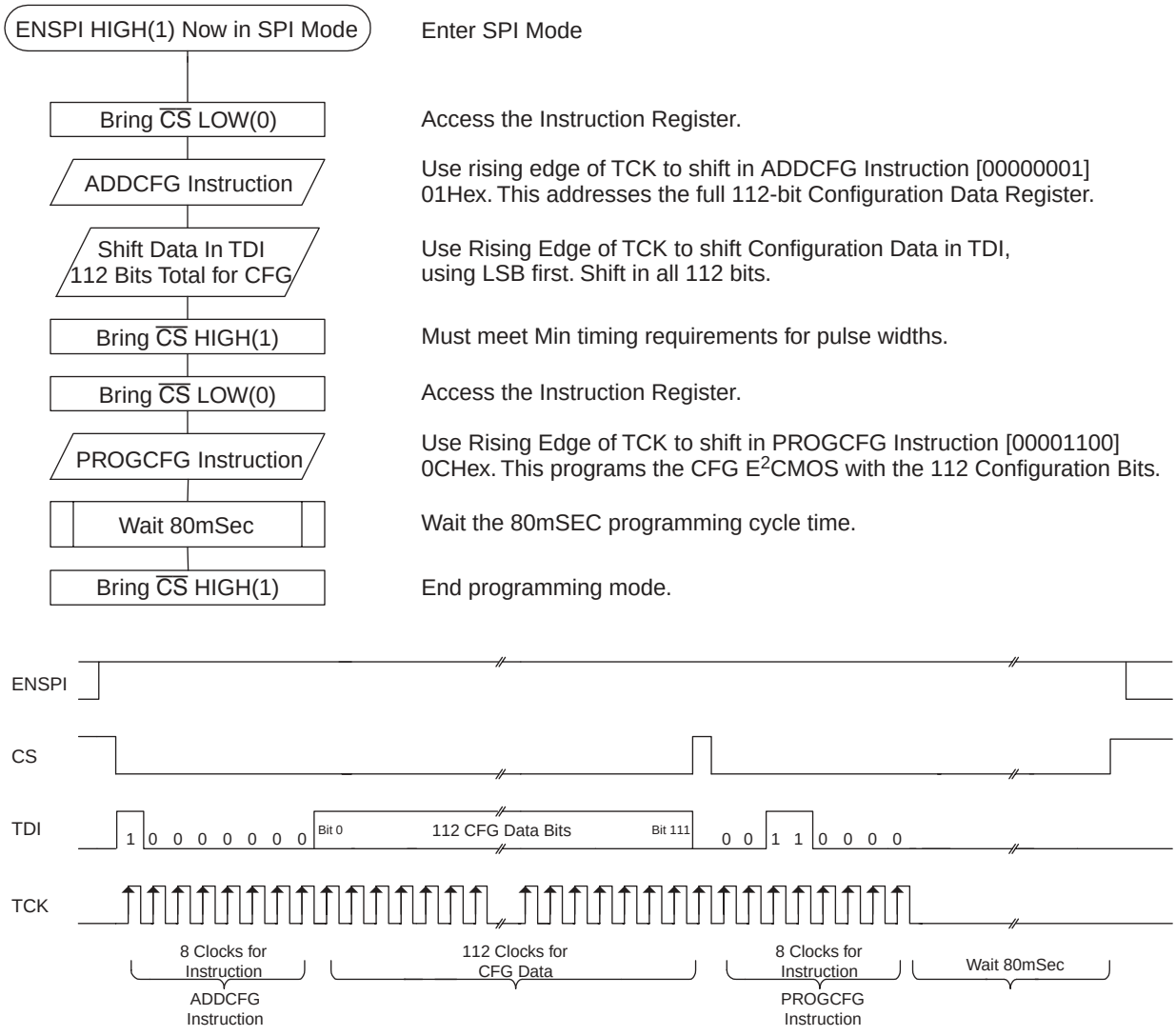




4.2.2. Write CFG

Writing to the E^2CMOS configuration bits is done to modify the non-volatile power-up-configuration of the ispPAC30. Writing to this register does not affect the configuration or behavior of the analog circuitry, with the exception of a minor increase in noise resulting from the cross coupling of digital signals. Before writing the 112 CFG bits, the E^2CMOS memory needs to be erased. During the programming cycle, $\overline{CS}$ should be held LOW for a minimum of 80 milliseconds after the PROGCFG instruction is shifted in. Reading after a write for verification should be delayed (as was discussed in the erase routines) by maintaining $\overline{CS}$ HIGH for a minimum of one millisecond before sending the READCFG instruction. The essential elements of writing the CFG bits into E^2CMOS are illustrated with a timing diagram in Figure 14. After the device is selected, the CFG register is addressed with the ADDCFG instruction followed by the 112 bits of CFG data. $\overline{CS}$ is brought HIGH then LOW and followed by the PROGCFG instruction. $\overline{CS}$ is maintained LOW for 80 milliseconds to meet the recommended programming time specified in the data sheet.

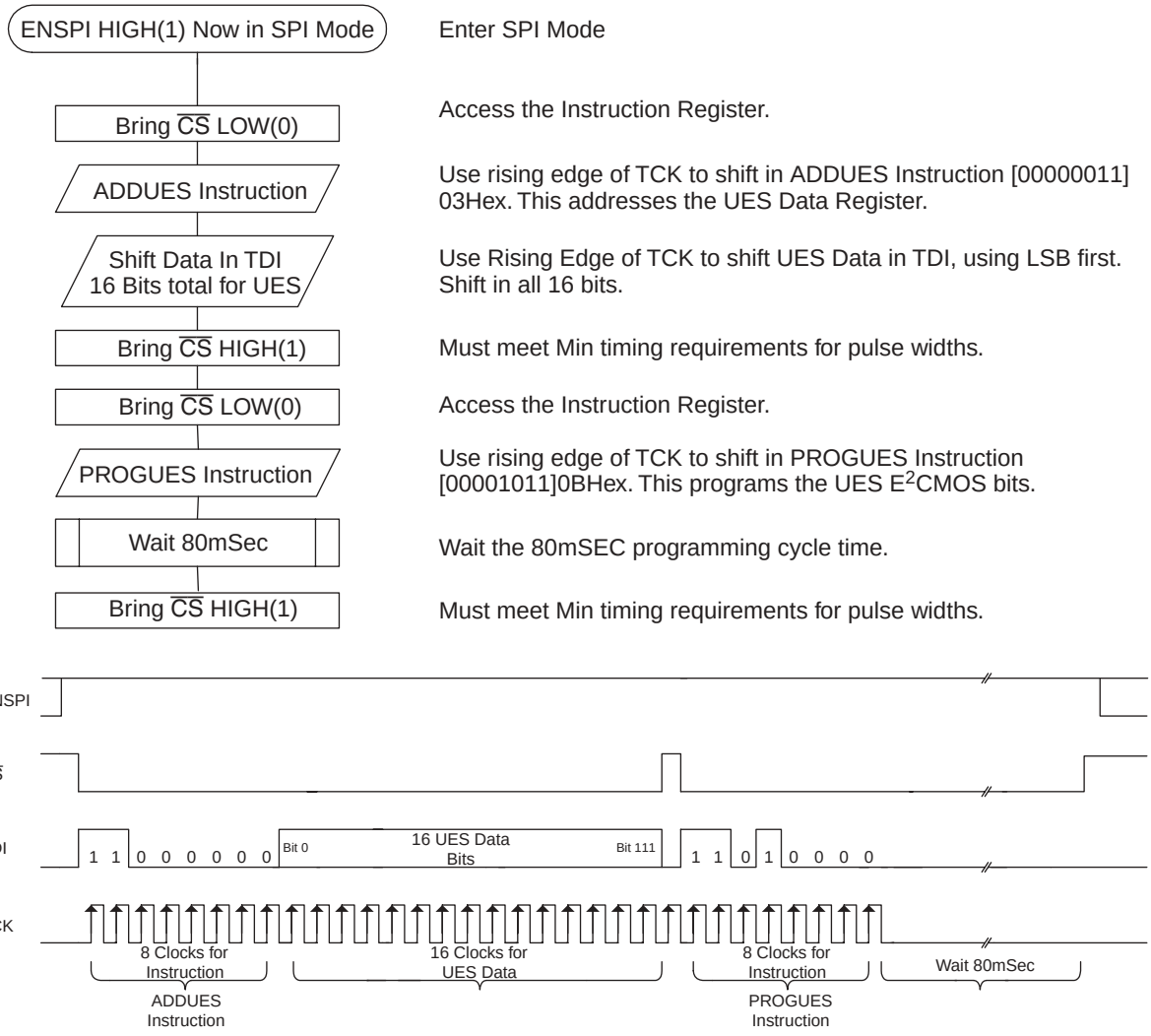
Figure 14. CFG Write Flow Chart and Timing Diagram



4.2.3. Write UES

This subroutine is used to modify the 16 non-volatile UES bits in E^2CMOS, which can be used to store board ID, manufacture date, look up data, etc. Before writing the UES bits, the E^2CMOS memory needs to be erased. During the programming cycle, $\overline{CS}$ should be held LOW for a minimum of 80 milliseconds after the PROGUES instruction is shifted in. Reading after a write for verification should be delayed (as was discussed in previous routines) by maintaining $\overline{CS}$ HIGH for a minimum of one millisecond before sending the READUES instruction. The write UES subroutine is outlined with a timing diagram in Figure 15. After the device is selected, the UES register is addressed with the ADDUES instruction followed by the 16 UES bits. $\overline{CS}$ is brought HIGH then LOW and followed by the PROGUES instruction. $\overline{CS}$ is maintained LOW for 80 milliseconds to meet the recommended programming time specified in the data sheet.

Figure 15. UES Write Flow Chart and Timing Diagram



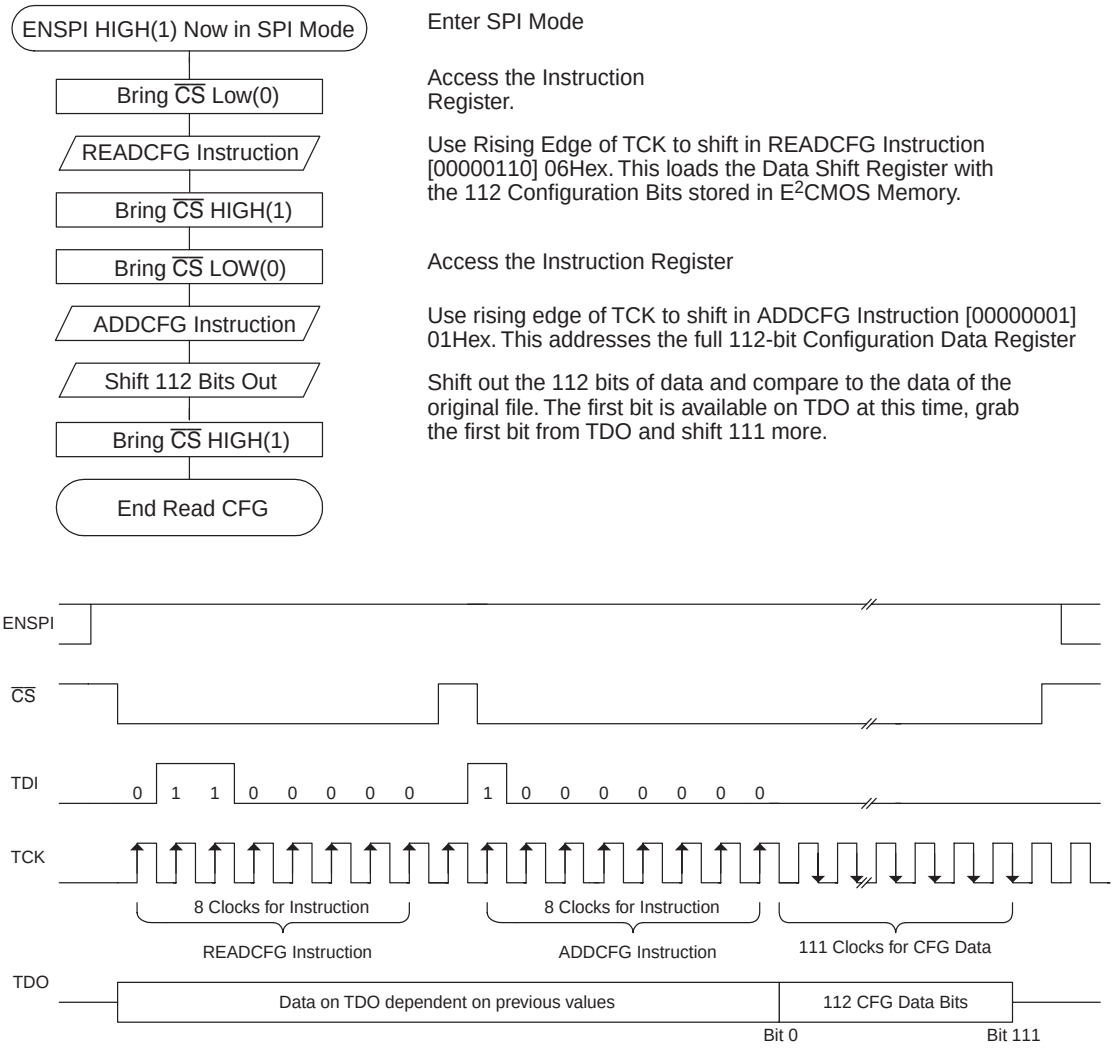
4.3. Read Routines

The reading subroutines are used to read the contents of the E^2CMOS registers for the 112 configuration bits, the 16 UES bits, and the 32 device ID bits. When a read routine immediately follows a write or erase routine, $\overline{CS}$ must remain HIGH for a minimum of one millisecond before it is brought LOW to shift in the respective read instruction. When shifting data out of the ispPAC30, it becomes valid after the falling clock edge (TCK), and the first bit is available on the output pin (TDO) after the last clock of the address instruction. This is illustrated in the following timing diagrams. Note the contents of the SRAM are not available for reading. It is recommended that a copy of the CFG bits be maintained by the system controlling the ispPAC30 for both continuity and performance.

4.3.1. Read CFG

After writing the 112 configuration bits, it is a good idea to read them back to verify the write process was successful. Figure 16 contains both a flow chart and timing diagram illustrating how to read the CFG bits. If a write or erase instruction immediately precedes the read, $\overline{CS}$ must remain HIGH for one millisecond before selecting the device for the read. After the device is selected, the READCFG instruction copies the data from E^2CMOS. $\overline{CS}$ is brought HIGH then LOW followed by the ADDCFG instruction. After the last clock of the ADDCFG instruction, the LSB of the data is available on the output pin (TDO). Thus, 111 more clock pulses are required to shift the 112 bits out.

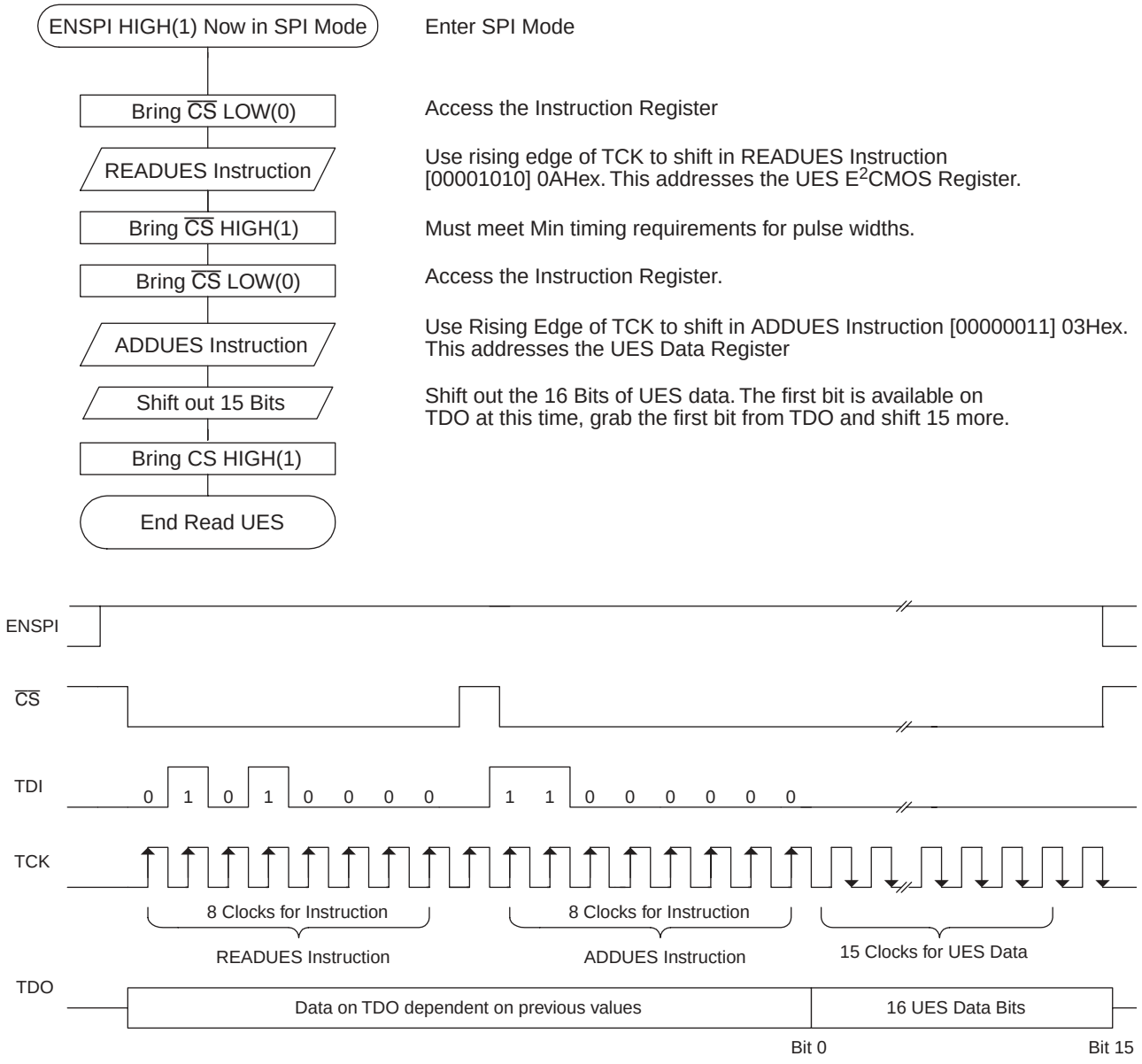
Figure 16. CFG Read Flow Chart and Timing Diagram



4.3.2. Read UES

Reading the UES bits may be done for any number of reasons including the following: verification of the write process, read-modify-write for counters, or read and report contents. Regardless of the reason to read the 16 UES bits, Figure 17 contains both a flow chart and timing diagram illustrating how to read the UES bits. If a write or erase instruction immediately precedes the read, $\overline{CS}$ must remain HIGH for one millisecond before selecting the device for the read. After the device is selected, the READUES instruction copies the data from E^2CMOS. $\overline{CS}$ is brought HIGH then LOW followed by the ADDUES instruction. After the last clock of the ADDUES instruction, the LSB of the data is available on the output pin (TDO). Thus, 15 more clock pulses are required to read all 16 bits.

Figure 17. UES Read Flow Chart and Timing Diagram



4.3.3. Read ID

IC programmers typically use the ispPAC30 ID code to verify the correct device is in the socket. Alternatively, board testers or embedded boot-up routines that configure In-System Programmable (ISP™) devices can also read the ID code before initiating a first-power-up programming sequence. Figure 18 contains both a flow chart and timing diagram illustrating how to read the device ID. After the device is selected, the IDCODE instruction connects the input (TDI) and output (TDO) pins to the ID register. After the last clock of the IDCODE instruction, the LSB of the data is available on the output pin (TDO). Thus, 31 more clock pulses are required to read all 32 bits. The 32 bits of the ID code are tabulated in Table 3 and they conform to the IEEE 1149.1 (JTAG) specification.

Figure 18. ID Read Flow Chart and Timing Diagram

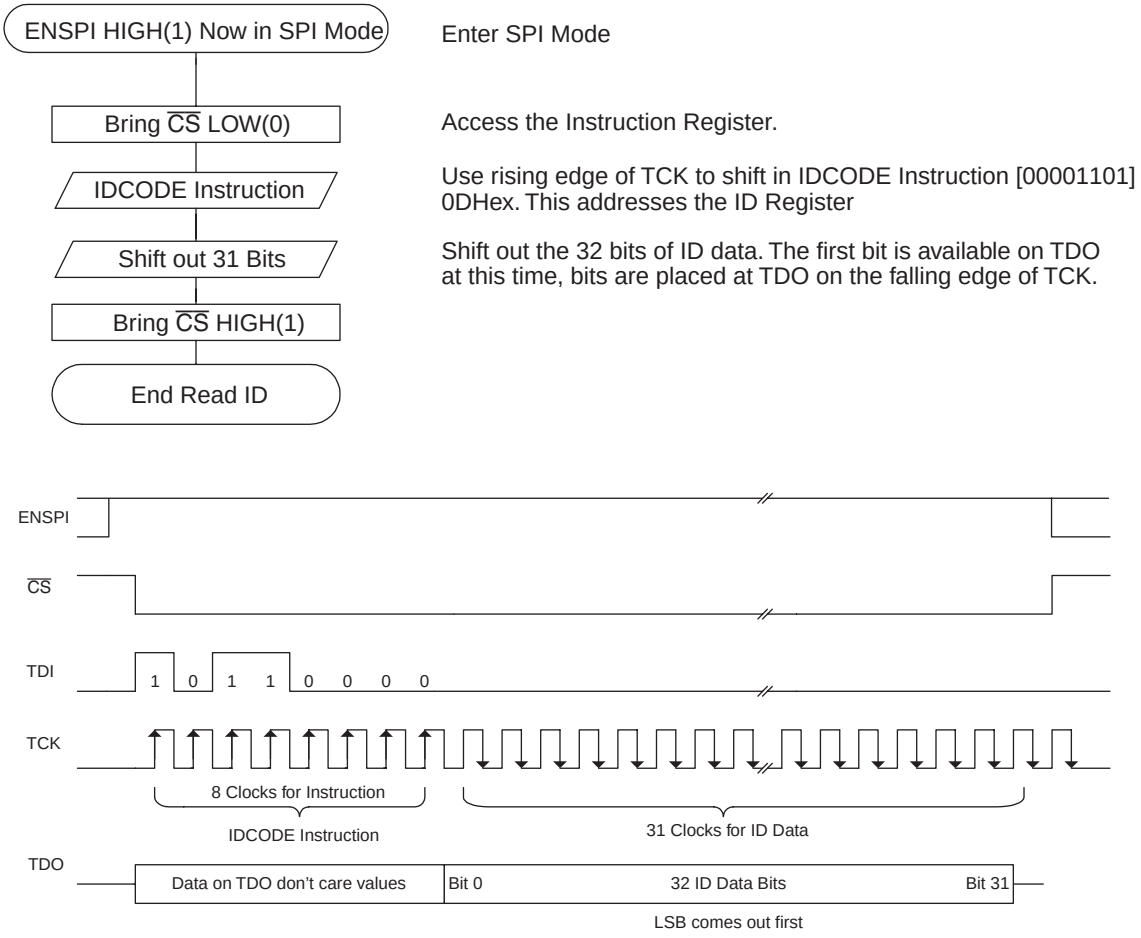


Table 3. ispPAC30 Device ID

Bit #	Value	Definition
0	1	Constant=1, per IEEE 1149.1 (JTAG) spec
1	1	Lattice Manufacturer ID
2	0	Lattice Manufacturer ID
3	0	Lattice Manufacturer ID
4	0	Lattice Manufacturer ID
5	0	Lattice Manufacturer ID
6	1	Lattice Manufacturer ID
7	0	Lattice Manufacturer ID
8	0	Lattice Manufacturer ID
9	0	Lattice Manufacturer ID
10	0	Lattice Manufacturer ID
11	0	Lattice Manufacturer ID
12	0	Part number
13	0	Part number
14	0	Part number
15	0	Part number
16	1	Part number
17	1	Part number
18	0	Part number
19	0	Part number
20	1	Part number
21	0	Part number
22	0	Part number
23	0	Part number
24	0	Part number
25	0	Part number
26	0	Part number
27	0	Part number
28	0	“Version”, programmable at factory, currently 0
29	0	“Version”, programmable at factory, currently 0
30	0	“Version”, programmable at factory, currently 0
31	0	“Version”, programmable at factory, currently 0

00130013_H

4.4. Power Down

This section describes the power up and power down instructions. They are both single instructions that do not involve data registers. Thus the flow charts and timing diagrams are quite simple: starting with selecting the device, followed by shifting the eight-bit instruction in, and ending with raising $\overline{CS}$. Simple as they are, they are included here for completeness. To conserve energy in battery powered systems; the analog circuitry can be powered down when not in use and powered up on demand. Typical power consumption during power down is around 10μA and the digital circuitry, that supports SPI, remains active at this level so that it can receive the POWERUP instruction. The flow charts and timing diagrams for the POWERUP and POWERDN instructions are detailed in Figures 19 and 20 respectively. The device will actually power up or down during the last clock cycle of the respective instruction. Alternatively the external pin $\overline{PD}$ or the individual CFG bits can be used to control the power to the analog circuits.

For further information regarding the power down CFG bits, see section V-H OA Configuration and Appendix B Configuration Bit Table.

Figure 19. Power-down Flow Chart and Timing Diagram

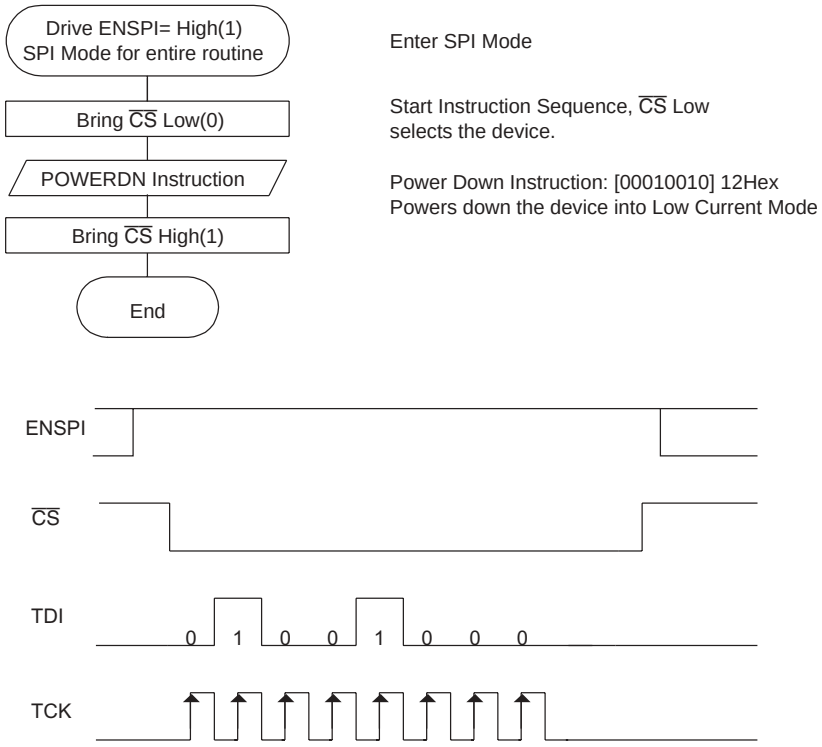
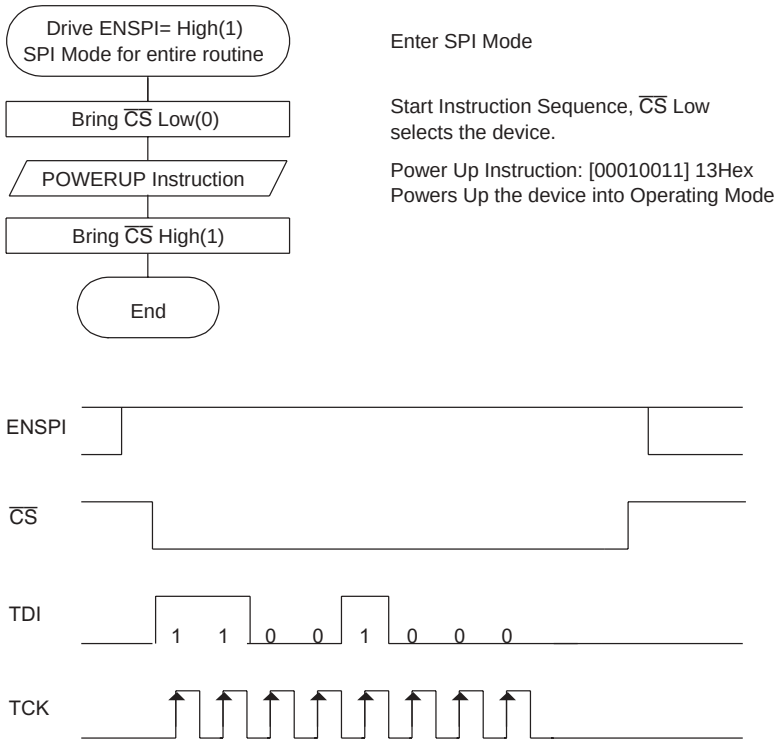


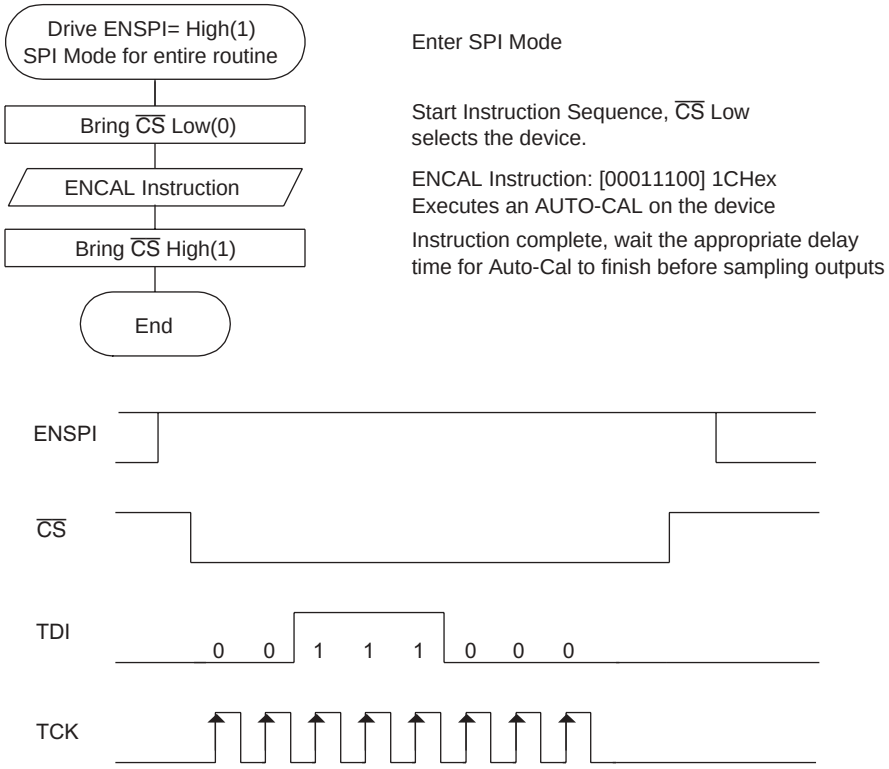
Figure 20. Power-up Flow Chart and Timing Diagram



4.5. Auto-calibration

This section describes the auto-calibration instruction. It is a single instruction that does not involve data registers. Thus the flow chart and timing diagram is quite simple: starting with selecting the device, followed by shifting the eight-bit instruction in, and ending with raising $\overline{CS}$. Simple as it is, it is included here for completeness. The ispPAC30 contains an auto-calibration (auto-cal) feature that can be initiated either from an external pin (CAL) or from an SPI instruction. Auto-cal re-routes internal circuitry to fixed references and then trims the circuits to minimize the offset as seen at the output. The process takes about 100 milliseconds. During this time, the outputs will be driven low and one should keep this in mind if a potential level shift or output low causes problems downstream. Thus, the outputs should to be ignored during an auto-cal sequence. The ENCAL instruction initiates during the last clock of the instruction. For additional details regarding the auto-cal feature, please refer to the ispPAC30 data sheet.

Figure 21. Auto-cal Flow Chart and Timing Diagram



Note: Outputs will remain invalid (low) for 100 milliseconds following the last clock of the ENCAL instruction.

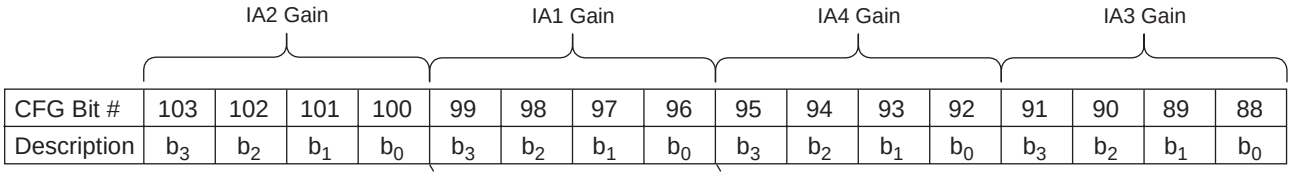
5. Configuration Bits

In this section the 112 configuration bits are described in detail within each of the functional groups; such as gain, polarity, and input routing, just to mention a few. Within each section, bit tables list only the valid configurations; thus bit combinations not listed in the tables are undefined and should not be used. Furthermore, performance and accuracy specifications are not guaranteed and circuit behavior may be unpredictable for any invalid bit combination not listed in the tables that follow.

5.1. IA Gain

The magnitude of the voltage gain for each of the IA (instrumentation amplifier) IA1, IA2, IA3, and IA4 is controlled by four bits each, for a total of 16 bits listed in Table 4. Note that a bit setting of “zero” results in a gain of one. As an example, Figure 22 shows IA2 set to a gain of 5 and the respective CFG bits.

Table 4. IA Gain Settings



CFG Bit #	103	102	101	100	99	98	97	96	95	94	93	92	91	90	89	88
Description	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0

b_3	b_2	b_1	b_0	Amplifier Gain
0	0	0	0	1
0	0	0	1	2
0	0	1	0	3
0	0	1	1	4
0	1	0	0	5
0	1	0	1	6
0	1	1	0	7
0	1	1	1	8
1	0	0	0	9
1	0	0	1	10

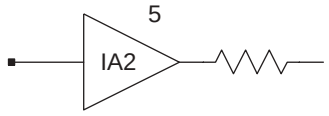
5.2. IA Polarity

The polarity bit controls the sign of the gain for each instrumentation amplifier (IA). When the polarity bit is LOW (0), the gain is positive and when it is HIGH (1) the gain is negative. Table 5 tabulates the bits that configure the polarity of the four IA's.

Table 5. Polarity Bits

Bit #	CFG register E^2 Bit Name	Description
82	IA3_POL	IA3 Polarity Control Bit: 0=Non-inverting, 1=Inverting
83	IA4_POL	IA4 Polarity Control Bit: 0=Non-inverting, 1=Inverting
86	IA1_POL	IA1 Polarity Control Bit: 0=Non-inverting, 1=Inverting
87	IA2_POL	IA2 Polarity Control Bit: 0=Non-inverting, 1=Inverting

Figure 22. Example: Set Gain of IA1 Equal to 5

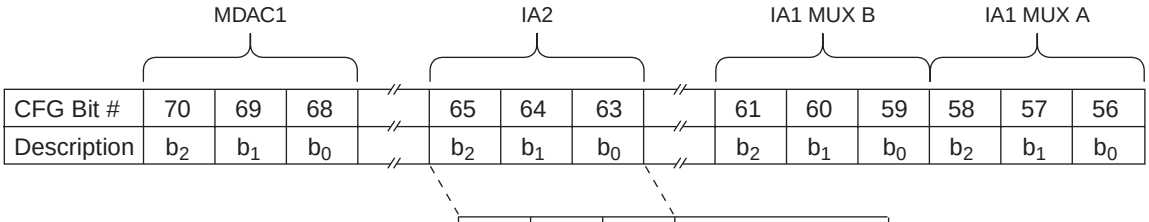


CFG Bit #	103	102	101	100		87
Value	0	1	0	0		0

5.3. Input Routing

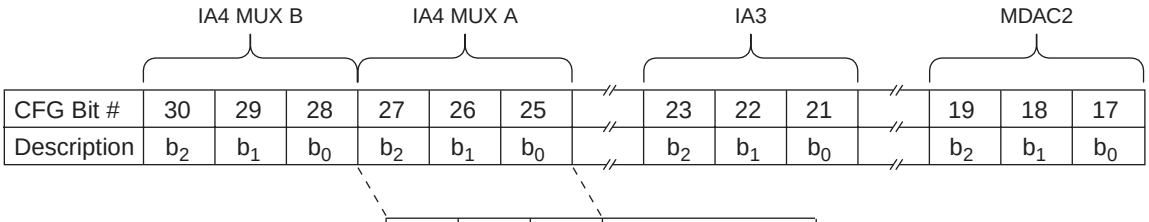
Three bits are required for each of the IAs and MDACs to configure their inputs. When the bits are set to zero, the inputs are essentially grounded and do not float. The input routing bits for the “upper” portion of the device (IA1, IA2 and MDAC1) are tabulated in Table 6, while the “lower” portion (MDAC2, IA3, IA4) is listed in Table 8. Please refer to the schematic screen of PAC-Designer or the ispPAC30 data sheet for further architectural clarification.

Table 6. Input Routing Bits and Settings, Upper



b_2	b_1	b_0	Input Connection
0	0	0	NONE
0	0	1	IN1
0	1	0	IN2
0	1	1	IN3
1	0	0	IN4
1	0	1	OUT1
1	1	0	OUT2
1	1	1	VREF1

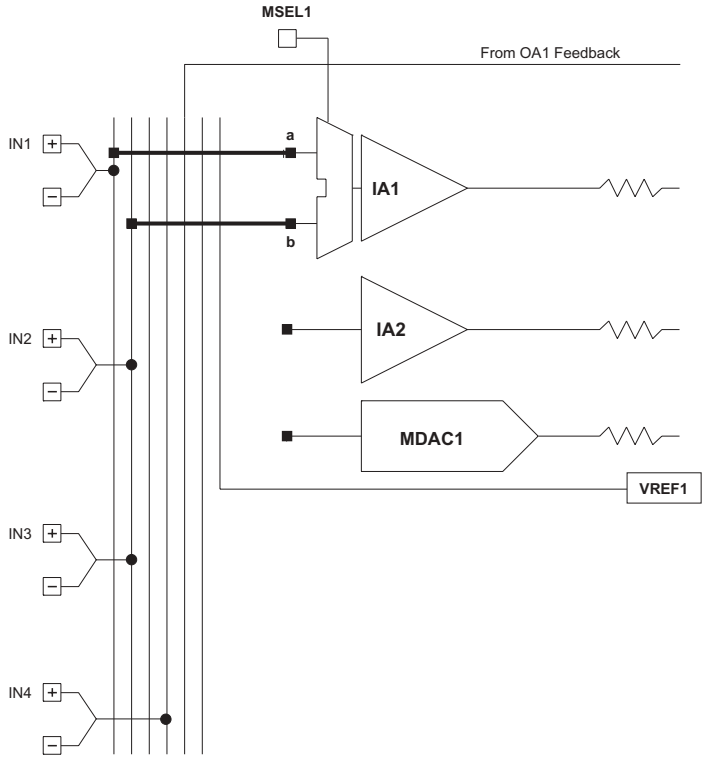
Table 7. Input Routing Bits and Settings, Lower



b_2	b_1	b_0	Input Connection
0	0	0	NONE
0	0	1	IN1
0	1	0	IN2
0	1	1	IN3
1	0	0	IN4
1	0	1	OUT1
1	1	0	OUT2
1	1	1	VREF2

Figure 23 indicates an interconnection between the MUX(a) input of IA1 to IN1 and MUX(b) input of IA1 to IN2. The MDAC1 and VREF1 are not used in this design example. Note the bits that control the routing for these inputs are set to zero.

Figure 23. Partial Schematic Showing Input Routing



	MDAC1				IA2				IA1 MUX B			IA1 MUX A		
CFG Bit #	70	69	68		65	64	63		61	60	59	58	57	56
Description	0	0	0		0	0	0		0	1	0	0	0	1

5.4. Input Multiplexers

The ispPAC30 provides real time flexibility; in that both IA1 and IA4 are equipped with a 2:1 analog multiplexer (MUX), which selects from internal nodes “A” or “B.” Each MUX is controlled by a two input exclusive-OR gate. One gate input is a bit from SRAM while the other input is an external pin (MSEL1, MSEL2). The encoding for the MUX control options are shown below in Tables 8 and 9. Figure 24 illustrates the internal SRAM and XOR gate that are used to select the MUX inputs for IA1 and IA4. Note: the schematic of ispPAC30 in PAC-Designer Version 1.30 identifies the MUX inputs with lower case “a” and “b.”

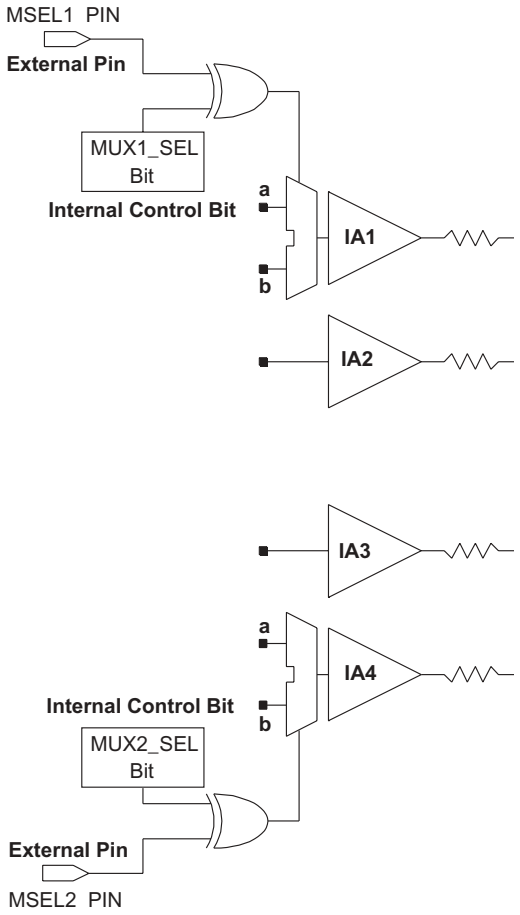
Table 8. MUX1 Control Logic

MUX1_SEL Bit # (85)	MSEL1 Pin State	Result
0	0	Selects Input (a) for MUX1
0	1	Selects Input (b) for MUX1
1	0	Selects Input (b) for MUX1
1	1	Selects Input (a) for MUX1

Table 9. MUX2 Control Logic

MUX2_SEL Bit # (84)	MSEL2 Pin State	Result
0	0	Selects Input (a) for MUX2
0	1	Selects Input (b) for MUX2
1	0	Selects Input (b) for MUX2
1	1	Selects Input (a) for MUX2

Figure 24. MUX1 and MUX2 Control Logic Schematic



5.5. Voltage References

The ispPAC30 has two adjustable Voltage references VREF1 and VREF2. Each can be set to fixed voltage settings based on the configuration bits listed in Table 10. The outputs of the Voltage references can be fed to the IA inputs or the MDACs. They provide flexibility in circuit configuration for fine tuning an offset with the 64 millivolt setting, or providing a DC offset in the output with the 2.5V setting. Note that VREF1 is only available to IA1, IA2, and MDAC1, while VREF2 is only available to IA3, IA4, and MDAC2.

Table 10. VFEF Bits and Settings

	VREF1			VREF2		
CFG Bit #	37	36	35	34	33	32
Description	b_2	b_1	b_0	b_2	b_1	b_0

b_2	b_1	b_0	Voltage Reference
0	0	0	64mV
0	0	1	128mV
0	1	0	256mV
0	1	1	512mV
1	0	0	1024mV
1	0	1	2048mV
1	1	0	2500mV

5.6. Multiplying DACs

The ispPAC30 is equipped with two multiplying DACs, MDAC1 and MDAC2. Each can multiply their respective inputs by a factor ranging from -1.0 to +0.9922. The range is divided into 256 equal steps, which enables fine tuning either a gain or offset when the input is from an external signal or internal VREF. An arbitrary waveform generator can be implemented when a VREF is connected to the MDAC input. When the quick configuration register (see Appendix C) is used to update the MDAC, the maximum sample rate is around 200 kHz. The source inputs for the MDACs include the four external differential inputs, the output amplifiers, and VREFs (see section C, Input Routing).

Table 11. MDAC Bits and Settings

	MDAC 1									MDAC 2							
CFG Bit #	111	110	109	108	107	106	105	104		79	78	77	76	75	74	73	72
Description	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0		b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	Decimal	MDAC Voltage Gain
0	0	0	0	0	0	0	0	0	-1.0000
0	0	0	0	0	0	0	1	1	-0.9922
0	0	0	0	0	0	1	0	2	-0.9844
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
0	1	1	1	1	1	1	1	127	-0.0078
1	0	0	0	0	0	0	0	128	0.0000
1	0	0	0	0	0	0	1	129	+0.0078
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
1	1	1	1	1	1	0	1	253	+0.9766
1	1	1	1	1	1	1	0	254	+0.9844
1	1	1	1	1	1	1	1	255	+0.9922

5.7. Summing Bus

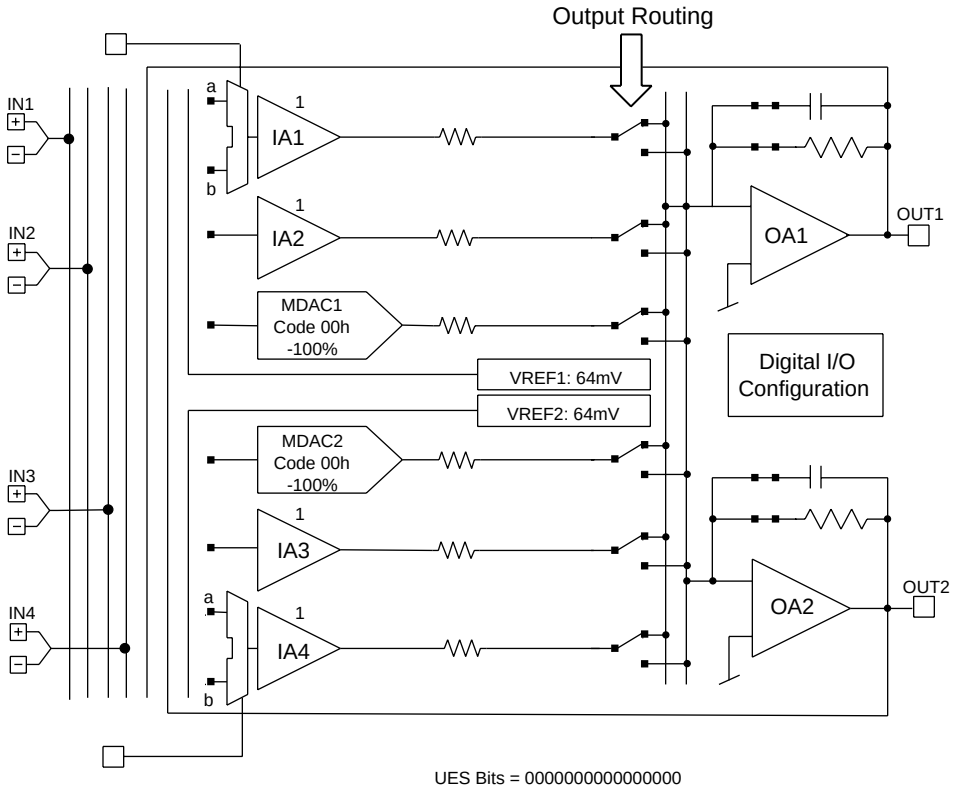
Table 12 shows the bit encoding for routing IA and MDAC outputs to the inputs of OA1 or OA2. The bit encoding is as follows: there are four IA's and two MDACs, there is a single bit for each IA output and one for each MDAC output. A value of (0) routes the signal to OA1, a value of (1) routes it to OA2.

Table 12. Summing Bus Bits and Settings

Bit #	CFG Bit
16	DAC2_OUT
38	IA4_OUT
39	IA3_OUT
54	IA1_OUT
55	IA2_OUT
67	DAC1_OUT

Bit Set To	Connects to Input
0	OA1
1	OA2

Figure 25. ispPAC30 Schematic (Arrow Depicts IA, MDAC Routing)



5.8. OA Configuration

The output amplifiers (OA) of the ispPAC30 are very versatile; they can function as a low pass filter, an integrator, or comparator. In low pass filter mode, one can change the (on chip) feedback capacitor to adjust the corner frequency of the OA. This is essentially a single order filter with a gentle roll off, for applications that require a steeper

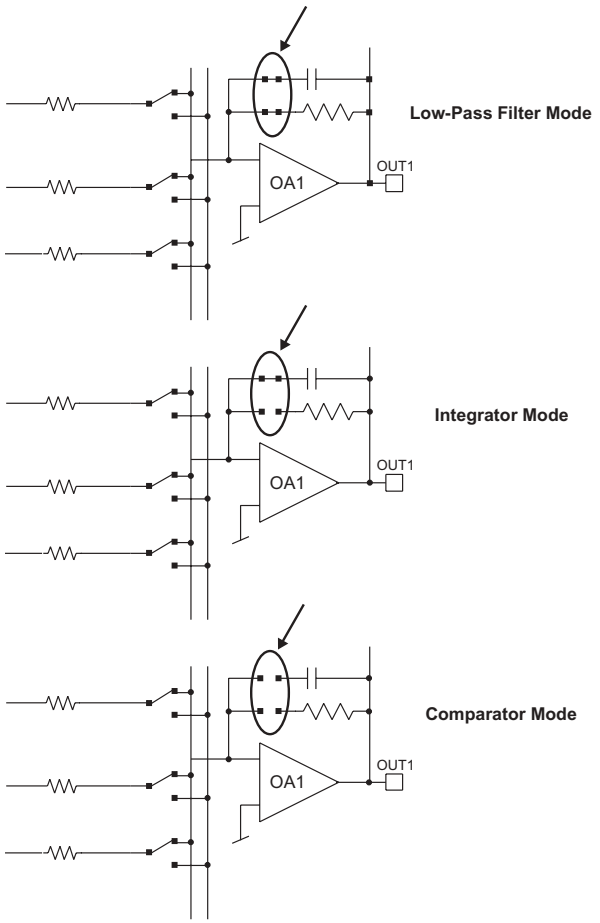
slope, please consider other ispPAC devices. In integrator mode the (internal) feedback resistor is removed from circuit and the feedback capacitor can be set to one of seven values to vary the integration time. The actual integration time constant is dependent on IA (or MDAC) gain and feedback capacitor value. In comparator mode, both (built in) feedback elements are removed from the circuit, and the slew rate of the OA is sped up to operate as a comparator. When the OA input (summing junction) is greater than zero, the output is HIGH and when the sum is less than zero (using negative gain and differential signals), the output is LOW. The summing junction is actually differential so negative sums can exist on a single supply device (see Lattice application note number AN6019 *Differential Signaling*). For additional information regarding OA configuration, please refer to the ispPAC30 data sheet. The configuration of each OA is controlled by two bits, as shown in Table 13 and illustrated in Figure 26.

Table 13. OA Configuration Bits and Settings

	OA 1			OA 2	
CFG Bit #	49	48		42	41
Description	b_1	b_0		b_1	b_0

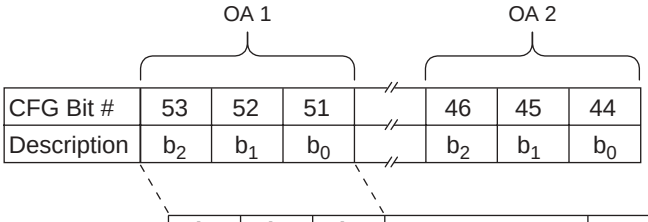
b_1	b_0	OA Configuration
0	0	Lowpass Filter
1	0	Integrator
1	1	Comparator

Figure 26. OA 1 Shown in the Three Different Modes



Each OA in the ispPAC30 has a capacitor in the feedback path, which is in parallel with a resistor. This capacitor is composed of a set of discrete capacitors in parallel that are switched into the circuit depending on the CFG data bits, as listed in Table 14.

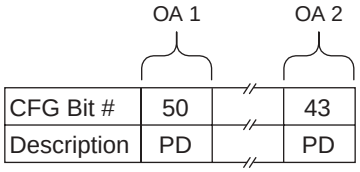
Table 14. OA Feedback Capacitor Bits and Settings



b_2	b_1	b_0	Capacitor Value	Corner Frequency
0	0	0	1.02pF	1.57MHz
0	0	1	4.32pF	619kHz
0	1	0	7.16pF	401kHz
0	1	1	11.97pF	250kHz
1	0	0	18.16pF	169kHz
1	0	1	27.29pF	114kHz
1	1	0	42.37pF	74kHz
1	1	1	64.01pF	49kHz

In addition to the POWERUP and POWERDN instructions that were discussed earlier, each OA can be powered up or down individually by clearing or setting the respective CFG bit. Thus the analog power budget can be fine-tuned based on the circuitry being used. Table 15 lists the bits that control the OA power and their settings.

Table 15. OA Power Mode Bits and Settings



PD	OA Power Mode
0	Power Up
1	Power Down

5.9. Digital Pins and Pull-up/down Settings

In addition to being able to reconfigure the analog circuitry, the ispPAC30 is also capable of modifying the behavior of the digital control pins. Several pins are equipped with a pull-up and a pull-down resistor. The resistors are relatively high impedance (~100kΩ); so that even if they are enabled, an external 10 kΩ can override them.

By changing the MUX select pin's pull-up/down with SPI, external digital I/O lines do not have to be used to select the MUX input.

Table 16. Internal Pull-up/down MUX1 Pin

MUX1 Pin	CFG Bit #10	CFG Bit #11
Pull-down	0	0
None	0	1
Pull-up	1	1

Table 17. Internal Pull-up/down MUX2 Pin

MUX2 Pin	CFG Bit #8	CFG Bit #9
Pull-down	0	0
None	0	1
Pull-up	1	1

While dynamically changing the pull up/down on the clock pin may not make much sense, it can be set to complement a clock source with an open drain structure.

Table 18. Internal Pull-up/down TCK Pin

TCK Pin	CFG Bit #12	CFG Bit #13
Pull-down	0	0
None	0	1
Pull-up	1	1

Table 19. Internal Pull-up/down Power-down Pin

PD Pin	CFG Bit #6	CFG Bit #7
Pull-up	0	0
None	1	0
Pull-down	1	1

Table 20. Internal Pull-up/down Enable SPI Pin

ENSPI Pin	CFG Bit #15
Pull-down	0
Pull-up	1

Table 21. TDO Pin Slew Rate

MUX1 Pin Slew Rate	CFG Bit #14
Slow	0
Fast	1

5.10. Auto-cal Voltage Settings

The ispPAC30 has internal circuitry that performs a voltage offset calibration in response to an ENCAL instruction as well as after every power-up event. The circuitry has an option to perform the calibration with a common mode voltage of 0V or at 2.5V. A single bit is used for each IA and each MDAC to select the common mode voltage used during calibration.

Table 22. Auto-cal Bits and Settings

Bit #	CFG Bit Name
20	DAC2_CALVOLT
24	IA3_CALVOLT
31	IA4_CALVOLT
62	IA1_CALVOLT
66	IA2_CALVOLT
71	DAC1_CALVOLT

Bit Set to	Definition
0	Auto-cal at 0V
1	Auto-cal at 2.5V

The default value for Auto-cal voltage settings are all set to (0), this will perform the auto-cal function with a reference of 0V. See the ispPAC30 data sheet for further information on Auto-cal parameters.

6. Summary

We have discussed both the programming routines and the configuration bits in great detail. The goal at this point is to apply the information presented thus far to develop code or software to control and configure the ispPAC30 on the fly. To further support this goal, the following appendices have been prepared as a summary or cheat sheet listing the instructions and configuration bits in one convenient location. Hopefully we have also shown, throughout the course of this application note, that the ispPAC30 is a very versatile analog building block, whose potential is only limited by the imagination of the engineers who wisely choose to use it.

Appendix A. ispPAC30 SPI Instructions

Table 23. ispPAC30 SPI Instruction List

Instruction Keyword	Description
ADDCFG	The “address CFG” command is used to address the full Configuration (CFG) register and shift serial data in and out of the ispPAC30.
ADDCFGQ	The “address CFGQ” command is used to address the CFGQ register and shift serial data in and out of the ispPAC30. The CFGQ register is a 40-bit subset of the full 112-bit CFG register. The remaining part of the CFG register is bypassed and its data is left unchanged.
ADDUES	This command is used to address the User Electronic Signature (UES) string and shift serial data in and out of the ispPAC30.
LATCHCFG	This command is used to update the data in the CFG string SRAM with the contents currently in the CFG Data Shift Register.
READCFG	This command is used to read the content of the CFG E^2 memory into the CFG Data Shift Registers before shifting the data out using the ADDCFG command.
READUES	This command is used to read the content of the UES E^2 memory into the UES Data Shift Registers before shifting the data out using the ADDUES command.
PROGUES	This command is used to program the content of the UES Data Shift Register into the UES E^2 memory.
IDCODE	This command is used to address the ID Data Shift Register and shifts out the ispPAC30 device ID code.
PROGCFG	This command is used to program the content of the CFG SRAM Shift Registers into the CFG E^2 memory.
PROGESF	This command is used to program the Electronic Security Fuse (ESF) Bit. Programming the ESF bit will disable READCFG user command. If ESF bit is programmed (esf=1) and the CFG String is addressed, data shifted out of the TDO pin will be the last data shifted in the CFG Data Shift Registers and not the status of the CFG E^2 memory.
POWERDN	This command is used to put the ispPAC30 in sleep mode (low power consumption mode). The digital part of the chip is not powered down.
POWERUP	This command is used to “wake” the ispPAC30 from sleep mode. This command will not wake the ispPAC30 if the PDb pin is low.
RELOADCFG	This command is used to load the values of the CFG E^2 memory into the SRAM.
ERASECFG	This command is used to erase the content of the CFG E^2 memory. If this data is loaded to the shift register and shifted out, all zeros should be observed at the TDO pin.
ERASEUES	This command is used to erase the content of the UES E^2 memory. If this data is loaded to the shift register and shifted out all zeros should be observed at the TDO pin.
ENCAL	This command is used to initiate the auto-calibration of the ispPAC30.
CFGBE	This command is used to erase the entire CFG memory and the ESF bit. Note that this is the only way to erase the ESF bit.
BYPASS	This command is used to put the ispPAC30 in BYPASS mode. In this mode, data is serially clocked through the ispPAC30 with one clock delay.

Table 24. ispPAC30 SPI Instruction Codes

Instruction (decimal)	Instruction (binary)	Instruction (Hex)	Instruction Name
0	00000000	00h	BYPASS
1	00000001	01h	ADDCFG
2	00000010	02h	ADDCFGQ
3	00000011	03h	ADDUES
4	00000100	04h	BYPASS
5	00000101	05h	LATCHCFG
6	00000110	06h	READCFG
7	00000111	07h	BYPASS
8	00001000	08h	BYPASS
9	00001001	09h	BYPASS
10	00001010	0Ah	READUES
11	00001011	0Bh	PROGUES
12	00001100	0Ch	PROGCFG
13	00001101	0Dh	IDCODE
14	00001110	0Eh	BYPASS
15	00001111	0Fh	BYPASS
16	00010000	10h	BYPASS
17	00010001	11h	PROGESF
18	00010010	12h	POWERDN
19	00010011	13h	POWERUP
20	00010100	14h	BYPASS
21	00010101	15h	BYPASS
22	00010110	16h	RELOADCFG
23	00010111	17h	ERASECFG
24	00011000	18h	BYPASS
25	00011001	19h	BYPASS
26	00011010	1Ah	BYPASS
27	00011011	1Bh	ERASEUES
28	00011100	1Ch	ENCAL
29	00011101	1Dh	CFGBE
30	00011110	1Eh	BYPASS
31	00011111	1Fh	BYPASS
63	00111111	FFh	BYPASS

Appendix B. Configuration Bits

Table 25. Configuration Bit Table

Bit #	CFG Bit Name	Description
0	Not used	Set to 0
1	Not used	Set to 0
2	Not used	Set to 0
3	Not used	Set to 0
4	Not used	Set to 0
5	Not used	Set to 0
6	PDB_pin_PU	Digital Pin Pull-up/down control
7	PDB_pin_PD	Digital Pin Pull-up/down control
8	MUX2_pin_PU	Digital Pin Pull-up/down control
9	MUX2_pin_PD	Digital Pin Pull-up/down control
10	MUX1_pin_PU	Digital Pin Pull-up/down control
11	MUX1_pin_PD	Digital Pin Pull-up/down control
12	TCK_pin_PU	Digital Pin Pull-up/down control
13	TCK_pin_PD	Digital Pin Pull-up/down control
14	TDO_pin_SR	Digital Pin Pull-up/down control
15	ENSPI_PU	Digital Pin Pull-up/down control
16	DAC2_OUT	DAC2_Output
17	DAC2_IEN_0	DAC2_Interconnect
18	DAC2_IEN_1	DAC2_Interconnect
19	DAC2_IEN_2	DAC2_Interconnect
20	DAC2_CALVOLT	DAC2_Auto-Cal_Voltage_Level
21	IA3_IEN_0	IA3_Interconnect
22	IA3_IEN_1	IA3_Interconnect
23	IA3_IEN_2	IA3_Interconnect
24	IA3_CALVOLT	IA3_Auto-Cal_Voltage_Level
25	IA4A_IEN_0	IA4A_Interconnect
26	IA4A_IEN_1	IA4A_Interconnect
27	IA4A_IEN_2	IA4A_Interconnect
28	IA4B_IEN_0	IA4B_Interconnect
29	IA4B_IEN_1	IA4B_Interconnect
30	IA4B_IEN_2	IA4B_Interconnect
31	IA4_CALVOLT	IA4_Auto-Cal_Voltage_Level
32	REF2_0	Voltage_Ref2
33	REF2_1	Voltage_Ref2
34	REF2_2	Voltage_Ref2
35	REF1_0	Voltage_Ref1
36	REF1_1	Voltage_Ref1
37	REF1_2	Voltage_Ref1
38	IA4_OUT	IA4_Output
39	IA3_OUT	IA3_Output
40	Not used	Set to 0
41	OA2_COMP	OA2_MODE

Bit #	CFG Bit Name	Description
42	OA2_INTEG	OA2_MODE
43	OA2_PD	OA2_Power_Mode
44	OA2_CAP_0	OA2_CAP
45	OA2_CAP_1	OA2_CAP
46	OA2_CAP_2	OA2_CAP
47	Not used	Set to 0
48	OA1_COMP	OA1_MODE
49	OA1_INTEG	OA1_MODE
50	OA1_PD	OA1_Power_Mode
51	OA1_CAP_0	OA1_CAP
52	OA1_CAP_1	OA1_CAP
53	OA1_CAP_2	OA1_CAP
54	IA1_OUT	IA1_Output
55	IA2_OUT	IA2_Output
56	IA1A_IEN_0	IA1A_Interconnect
57	IA1A_IEN_1	IA1A_Interconnect
58	IA1A_IEN_2	IA1A_Interconnect
59	IA1B_IEN_0	IA1B_Interconnect
60	IA1B_IEN_1	IA1B_Interconnect
61	IA1B_IEN_2	IA1B_Interconnect
62	IA1_CALVOLT	IA1_Auto-Cal_Voltage_Level
63	IA2_IEN_0	IA2_Interconnect
64	IA2_IEN_1	IA2_Interconnect
65	IA2_IEN_2	IA2_Interconnect
66	IA2_CALVOLT	IA2_Auto-Cal_Voltage_Level
67	DAC1_OUT	DAC1_Output
68	DAC1_IEN_0	DAC1_Interconnect
69	DAC1_IEN_1	DAC1_Interconnect
70	DAC1_IEN_2	DAC1_Interconnect
71	DAC1_CALVOLT	DAC1_Auto-Cal_Voltage_Level
72	DAC2_0	DAC2_Value
73	DAC2_1	DAC2_Value
74	DAC2_2	DAC2_Value
75	DAC2_3	DAC2_Value
76	DAC2_4	DAC2_Value
77	DAC2_5	DAC2_Value
78	DAC2_6	DAC2_Value
79	DAC2_7	DAC2_Value
80	DONTCARE_1	Not used
81	DONTCARE_2	Not used
82	IA3_POL	IA3_Polarity
83	IA4_POL	IA4_Polarity
84	MUX2_SEL	IA4_Input_MUX
85	MUX1_SEL	IA1_Input_MUX
86	IA1_POL	IA1_Polarity

Bit #	CFG Bit Name	Description
87	IA2_POL	IA2_Polarity
88	IA3_GAIN_0	IA3_Gain
89	IA3_GAIN_1	IA3_Gain
90	IA3_GAIN_2	IA3_Gain
91	IA3_GAIN_3	IA3_Gain
92	IA4_GAIN_0	IA4_Gain
93	IA4_GAIN_1	IA4_Gain
94	IA4_GAIN_2	IA4_Gain
95	IA4_GAIN_3	IA4_Gain
96	IA1_GAIN_0	IA1_Gain
97	IA1_GAIN_1	IA1_Gain
98	IA1_GAIN_2	IA1_Gain
99	IA1_GAIN_3	IA1_Gain
100	IA2_GAIN_0	IA2_Gain
101	IA2_GAIN_1	IA2_Gain
102	IA2_GAIN_2	IA2_Gain
103	IA2_GAIN_3	IA2_Gain
104	DAC1_0	DAC1_Value
105	DAC1_1	DAC1_Value
106	DAC1_2	DAC1_Value
107	DAC1_3	DAC1_Value
108	DAC1_4	DAC1_Value
109	DAC1_5	DAC1_Value
110	DAC1_6	DAC1_Value
111	DAC1_7	DAC1_Value

Appendix C. Quick Configuration Register

To update or change the configuration of the ispPAC30 quickly, a subset of the 112 configuration bits exists. This subset of forty bits provides control of the IA gains and MDAC settings, which are the sections most likely to be changed on the fly. For example: one could modulate an input signal with the MDAC, by programming it with an arbitrary waveform. It is important to remember to set ALL of the 112 configuration bits before subsequent writes to the quick configuration register are performed. This can be done by either a direct write, or having written to the E2CMOS and performing a READCFG instruction to set them. Failure to set all 112 bits before issuing the LATCHCFG instruction will probably not result in the desired output. Similar to writing to the other registers, the LSB or bit zero is shifted in first as illustrated in Figure x.

Figure 27. Quick Configuration Register Writing Flow Chart and Timing Diagram

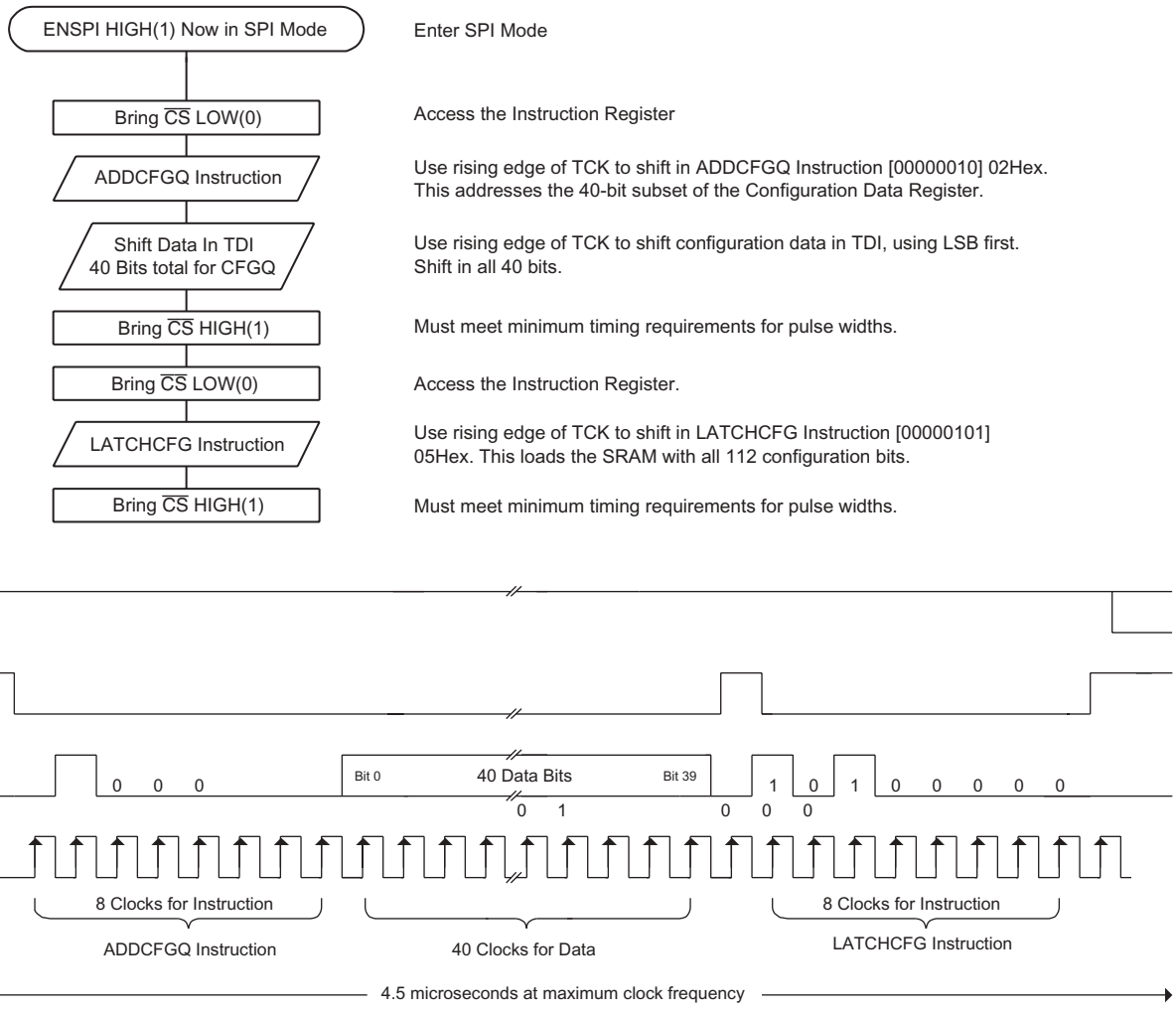


Table 26. Bit Map From Quick to Configuration Registers

Quick CFG Bit #	Full CFG Bit #	CFG Bit Name
0	72	DAC2_0
1	73	DAC2_1
2	74	DAC2_2
3	75	DAC2_3
4	76	DAC2_4
5	77	DAC2_5
6	78	DAC2_6
7	79	DAC2_7
8	80	DONTCARE_1
9	81	DONTCARE_2
10	82	IA3_POL
11	83	IA4_POL
12	84	MUX2_SEL
13	85	MUX1_SEL
14	86	IA1_POL
15	87	IA2_POL
16	88	IA3_GAIN_0
17	89	IA3_GAIN_1
18	90	IA3_GAIN_2
19	91	IA3_GAIN_3
20	92	IA4_GAIN_0
21	93	IA4_GAIN_1
22	94	IA4_GAIN_2
23	95	IA4_GAIN_3
24	96	IA1_GAIN_0
25	97	IA1_GAIN_1
26	98	IA1_GAIN_2
27	99	IA1_GAIN_3
28	100	IA2_GAIN_0
29	101	IA2_GAIN_1
30	102	IA2_GAIN_2
31	103	IA2_GAIN_3
32	104	DAC1_0
33	105	DAC1_1
34	106	DAC1_2
35	107	DAC1_3
36	108	DAC1_4
37	109	DAC1_5
38	110	DAC1_6
39	111	DAC1_7

Appendix D. JEDEC Bit Stream

Building the Bit Stream

The ispPAC30 contains 112 bits in the CFG (configuration register), as well as 16 bits in the UES register. These bits are all addressable and can be changed by the user. To design a system that incorporates the ispPAC30 in a reconfigurable environment, a good understanding of the data flow and data bit order is required. There are several ways to approach the task. The easiest method to configure or set the data bits in the string is to use Lattice PAC-Designer software to export a JEDEC file. The JEDEC file contains all bit information about the configuration for the E^2CMOS cells. The JEDEC file below shows 128 total bits, (112 CFG bits + 16 UES). Based on the JEDEC standard, the “QF128” defines the number of fuses or bits. The “QP28” defines the number of pins on the device. The first address starts at “L0000”, the first data bit is after “L0000”; this first bit is at address location (0) or the LSB of the CFG Register Data. The 16 bits of the UES data are located at the end of the JEDEC file.

Figure 28. JEDEC File Format

```
PAC-Designer  Design
Title:
Author: Information Systems
*
QF128*
QP28*
L0000
00000000000000001111001101100010000000110000111000000100100010010100111
00101101000110000010001101000110010100101111011000100111 0*
C05AF*                                                    5 A 7 3 2 7
3249                                                      MDAC1  UES Bits
```

JEDEC File Export

To begin the design process, use PAC-Designer to define the analog functions within the ispPAC30. Complete the routing and configuration of the device using the graphical interface. Once the schematic is defined, with routing, gains, MDAC and reference settings, the JEDEC file can be exported. This will provide the user with a template to work from. The Export function is located under the “File” menu in the PAC-Designer toolbar (Figure 30).

With the JEDEC file containing the full configuration information in its raw data form, the user can now format the data for a given storage medium such as in byte form to be accessed by the microprocessor or test platform or in a data file format. Multiple JEDEC files can also be used to define different set-ups or filter configurations. To program a different design into the ispPAC30, the system can simply use the stored data file that represents a given design.

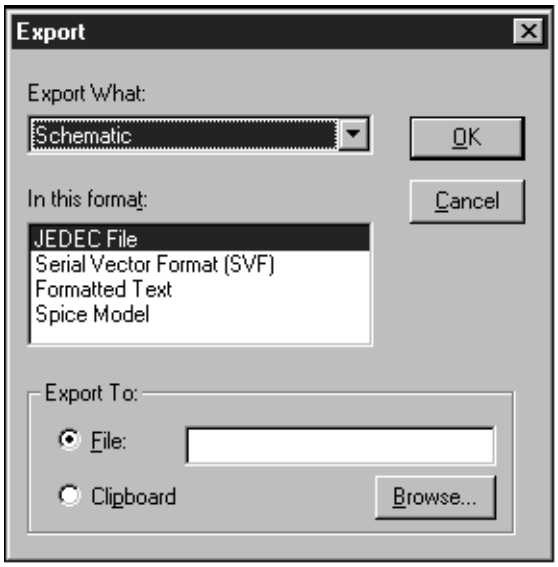
JEDEC File Conversion

Utilities are also available to convert the JEDEC file to a "C" or assembly data array that preserves the bit order. Figure 29 shows an example of a "C" code conversion. In this example, MDAC1 was set to 0xA5 and MDAC2 was set to 0x5A. See how they fall on even byte boundaries. Notice the bit reversal from JEDEC to "C," so the embedded programming routine can shift the least significant bit out first and proceed from the low order byte (pPAC_30[0]) to the high order byte (pPAC_30[15]).

Figure 29. “C” Data Array Generated from JEDEC File

```
unsigned char pPAC_30[] = {
0x00, 0x00, 0xCF, 0x46, 0xC0, 0x70, 0x20, 0x91, 0x72, 0x5A,
0x0C, 0x62, 0x31, 0xA5, 0x37, 0x72                        MDAC2
};
                  MDAC1  UES Bits
```

Figure 30. PAC-Designer Export Dialog



There are several methods that can be used to reconfigure the ispPAC30. As described above, the user predefines the JEDEC file and stores it as data to be downloaded via SPI. An alternative method is to use a predefined starting point template such a simple filter configuration with the proper routing already in place, then to dynamically edit the necessary bit locations that affect gain or MDAC settings, etc. The third way is to construct the full configuration bit pattern from the tables and documents. Using the JEDEC file is the most straightforward method. Note the JEDEC file format starts with the LSB as the first bit in the data. If the data is converted to byte form, the bit order must be reversed so that the respective data position is preserved when downloading.

Appendix E. Example Source Code

As part of Lattice's commitment to support ispPAC customers, the Applications Engineering department provides example code for a variety of microprocessor or DSP platforms for embedded or real time device control. The modules currently available have been compiled and tested in our labs. Each module consists of a library of functions that interface to the hardware and a memory image of the CFG bits.

The default modules are designed to interface through the PC parallel port to the evaluation board via the ispDOWNLOAD® cable. A simple modification to the ispDOWNLOAD cable will allow both SPI and JTAG programming. Use the "plug" wire (pin 5) to control CS, and an onboard jumper to enable/disable the SPI mode. Thus, it is possible to integrate the example module with a couple of function calls to program or modify an ispPAC device. Visit the Lattice web site at www.latticesemi.com to download code or contact Lattice ispPAC Applications Engineering and let us know what your requirements are.

Technical Support Assistance

Hotline: 1-800-LATTICE (Domestic)
1-408-826-6002 (International)
e-mail: ispPACs@latticesemi.com