ATW300 Family
Programming Guide

Introduction

This document is the Programmer's Guide for the Atrua family of Adaptive Capacitive touch sensors. Electrical specifications, physical dimensions and pin descriptions are contained in the ATW300 Family Datasheet, which is available from Atrua upon request.

ATW300 family sensors are high performance, low power, low cost sensors with an integrated 124 x 8 sensing array of metal electrodes utilizing Atrua's Adaptive Capacitive sensing technology. Each electrode acts as one plate of a capacitor, while the contacting finger acts as the second plate. An insulating layer on the device surface forms the dielectric between the two plates. Ridges and valleys on the finger yield varying capacitor values across the array, which is read to form a partial image of the fingerprint.

Internal circuits within ATW300 family of sensors convert the sensed data into a stream of digital data (a *frame*) that is presented to the host microprocessor via an 8-bit bi-directional bus interface compatible with most microprocessors, or via a high-speed synchronous serial interface compatible with SPI™. Processing algorithms running on the host perform image reconstruction by assembling the frames that are delivered as the finger is moved across the sensor array. This data is used to determine X-Y movement, rotation and pressure for the navigation and control algorithms. Other algorithms perform minutiae extraction and matching to form and compare templates that are stored and used for fingerprint authentication.

ATW300 sensors include the analog-to-digital converters necessary to digitize the sensed data, an automatic finger detection (AFD) circuit that sends an interrupt signal to the host microprocessor when a finger is placed on the sensor, and an automatic gain control (AGC) function that provides high quality fingerprint images from all types of skin, dry to moist, in a wide range of climatic conditions. The ATW300 sensors also integrate all recommended external components except for a few small resistors and capacitors.

The AFD interrupt allows the host microprocessor to remain in standby mode until a finger is placed on the sensor, eliminating the need for the host CPU to continually poll the fingerprint sensor to determine whether a finger is present. The interrupt function also permits the host to place the sensor into standby mode, dramatically reducing operating current. The current can be even further reduced by placing the device in power-down mode. The AGC function widens the application range of the sensor and reduces the false acceptance rate (FAR) and false rejection rate (FRR) for the fingerprint recognition function.

Conventions

Unless otherwise noted, a positive logic (active High) convention is assumed throughout this document. A lowercase 'n' following a signal name (e.g., INTRn) indicates that the signal is active Low.

Fixed-point values containing fractional parts are notated in "n.m" format, where n is the number of bits of the integer part, and m is the number of bits of the fractional part. For example, a 12-bit value consisting of a 4-bit integer part in the most significant bits and an 8-bit fractional part (resolving to 1/256th) in the least significant bits would be described as 4.8 format, for short.

The designation 0xNNNN indicates a hexadecimal number.

The designation 0bNNNN indicates a binary number.

SPI™ is a trademark of Motorola, Inc.

Sensor Operation

Figure 1 is a block diagram of the ATW300 sensor, showing the major internal blocks. In addition to the bus interface logic, the registers and the state controller, the chip contains a *sensor array*, an *analog-to-digital converter* and a *frame buffer*.

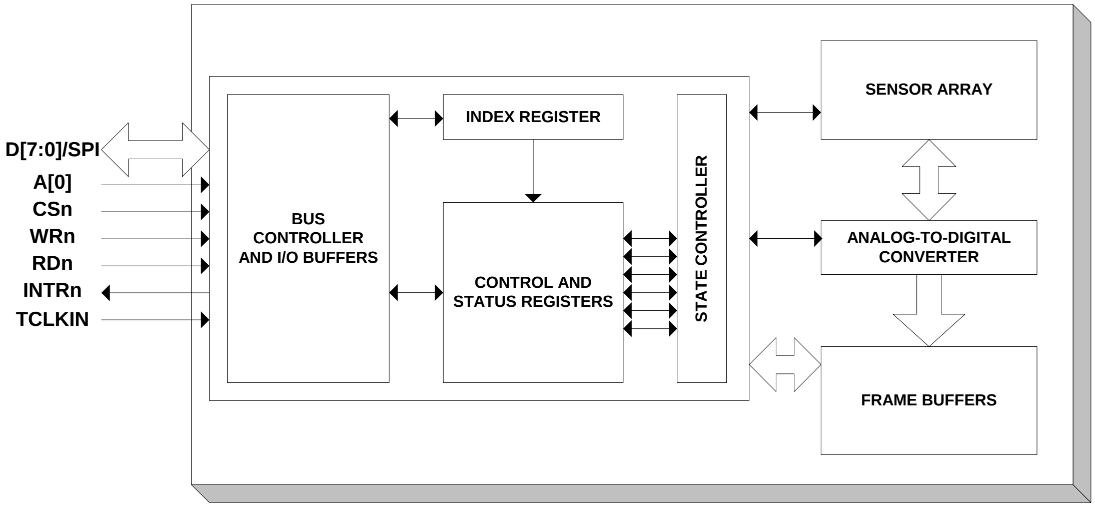


Figure 1. ATW300 block diagram

Sensor Array

The sensor consists of an array of 992 metal electrode pixels arranged in a matrix of 124 columns by 8 rows. Each electrode forms one side of a capacitor, with the finger placed on the sensor forming the second electrode.

Analog-to-Digital Converter

When a finger is placed on the sensor array, a capacitance is formed by each electrode and the finger. Valleys and ridges on the finger will yield varying capacitance values. All 124 columns (pixels) in a single row of the sensor array are measured simultaneously, with the process being repeated eight times to capture the data for the eight rows of pixels in the sensor array.

The capacitance for each pixel is digitized into a four-bit value ranging from 0x00 to 0x0F, with 0x00 representing black and 0x0F representing white. The data for two adjacent pixels is formed into a byte, with the data for the even pixel (columns 0, 2, 4, … 122) in the lower nibble of the byte and the data for the odd pixel (columns 1, 3, 5, … 123) in the upper nibble.

Frame Buffer

The frame buffer provides temporary storage for the digitized sensor data. When data is available to be read, a data available bit in one of the device's status registers is asserted. The host can then read the data for that frame by reading from the Data Register. Each read of that register automatically increments the read pointer for the data buffer, so that the data is read sequentially, starting with the byte containing the data for row 0, columns 0 and 1, continuing to the end of that row (row 0, columns 122 and 123), and then proceeding to the next row, until all data for that frame of the finger swipe has been output to the host. When a complete frame has been read, the data available bit will be cleared.

Standby and Power-down Modes

ATW300 offers two power saving modes, *standby mode* and *power-down mode*. Standby mode turns off the majority of the operating circuits within the sensor, but allows the surface contact detection function to operate. This mode can be invoked to reduce the operating current while waiting for the finger to be placed on the sensor's surface.To invoke the standby (partial power down) mode, the partial power down bit P_PWR_DN[6] in the IOC_REG should be set. In this mode, sensor registers cannot be read, and only certain registers can be written. Power-down mode consumes the least power and can be invoked when operation of the sensor is not required. To enter in power-down mode, the full power down bit F_PWR_DN[7] in the IOC_REG should be set.

Power-down mode is invoked when CMD_REG is set to 0x00 and the timestamp timer bit SCLK_EN, CLK0_REG[6] is cleared.

When the CMD_REG[2] bit is set, the device operates in the standby mode, and it remains in that mode until a surface contact is detected. If AUTO_RUN_EN, CMD_REG[4] is also set at the same time, the device will then automatically enter the normal operating mode and begin calibration and fingerprint data acquisition once a surface contact is detected. If AUTO_RUN_EN is not set, the sensor will remain in standby mode until the host specifically invokes an operating function by setting appropriate bits in the Command Register.

Bus Operations

Communications between the host processor and the ATW300 sensor are executed over one of two interfaces: an 8-bit bidirectional bus, using read, write and chip select control signals that are compatible with most microprocessors, or an SPI™-compatible synchronous serial interface. If interrupts are enabled, the occurrence of certain events is signaled to the host via an interrupt output signal.

8-bit Bidirectional Bus Interface

Table 1 describes the possible bus operations for the 8-bit bidirectional bus interface. A single address line (A0) is used to select the register for the read or write transaction. When A0 is Low, the Index register is selected. When A0 is High, one of the control or status registers is selected, using the current contents of the Index register as a pointer. Thus, two cycles are required to access a specific register for the transaction. In the first cycle, the register's pointer value should be written into the Index Register and then the required transaction with the desired register is performed during the second cycle.

Table 1. ATW300 8-bit Bidirectional Bus Operations[1]

Operation	*CSn*	*RDn*	*WRn[2]*	*A0*	*D[7:0]*
Deselected	H	X	X	X	Hi-Z
Selected/Idle	L	H	H	X	Hi-Z
Read Index Register	L	L	H	L	D_{OUT}
Write Index Register	L	H	L	L	D_{IN}
Read Data/Control/Status Register[3]	L	L	H	H	D_{OUT}
Write Data/Control/Status Register[3]	L	H	L	H	D_{IN}

Notes:
1. L = VIL, H = VIH, X = Don't Care. See "ATW300 Family Data Sheet" for Voltage Levels.
2. Data is latched on the rising (trailing) edge of WRn.
3. The location read from or written to is the register pointed to by the contents of the Index Register.

Serial Peripheral Interface (SPI)

In the Serial Peripheral Interface (SPI) mode the sensor functions as a slave device. The SPI interface uses four signals, SCSn, SCK, MOSI, and MISO.

Signal	Description
SCSn	Slave Chip Select. Active-low input to slave. The SPI master drives SCSn low to initiate a read transaction or write transaction. The SPI master drives SCSn high to terminate a read or write transaction.
SCK	Slave Clock. Input to slave. SCK is driven by the SPI master. Data as shifted out of the master or slave on the falling edge of SCK. Data is sampled by the master or slave on the rising edge of SCK.
MOSI	Master Output Slave Input. The MOSI pin is driven by the SPI master,

	which shifts data out to the slave device on the falling edge of SCK. The slave device samples the input data on the rising edge of SCK. Data is shifted out from the master most-significant-bit (MSB) first and LSB last.
MISO	Master Input Slave Output. The MISO pin is driven by the SPI slave, which shifts the data out to the master on the falling edge of SCK. The SPI master samples data from the slave device on the rising edge of SCK. Data is shifted out from the slave MSB first and LSB last.

There are two types of data transfers–Write Register and Read Register. A data transfer consists of two phases–a Command Phase and Data Phase. During the Command Phase, the SPI master sends an 8-bit command, which has two fields. Bits [7:1] specify the register address to be read or written. Bit 0 specifies whether the data transaction will write or read a register. If a register is to be written, the SPI master sends an 8-bit value during the data phase. If a register is to be read, then the SPI slave will send the contents of the register that was addressed.

Table 2. SPI Bus Operations

Command Bit	*Field*	*Description*
[7:1]	Register Address [6:0]	Specifies the register address to be read or written.
[0]	Read/Write	Specifies whether the command is a Read or Write command. 0=Write to the specified register. 1=Read from the specified register.

Register Write Procedure

To write a register, the SPI master (i.e., host controller) drives SCSn low, sends two bytes to the slave, and then drives SCSn high. The first byte is the Write Command and the second byte is the data to be written into the specified register. Refer to Figure 2.

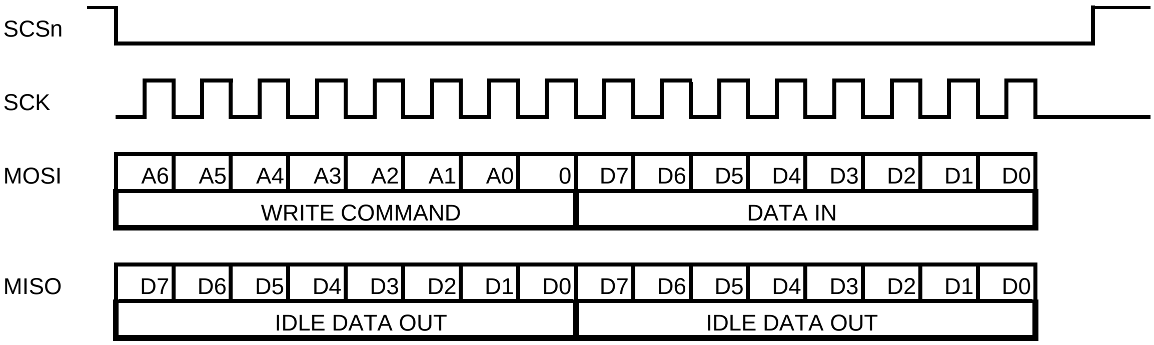


Figure 2. SPI Single-Byte Write

By holding SCSn low, the master can continue to write to the slave by sending the additional Command/Data byte pairs. Refer to Figure 3.

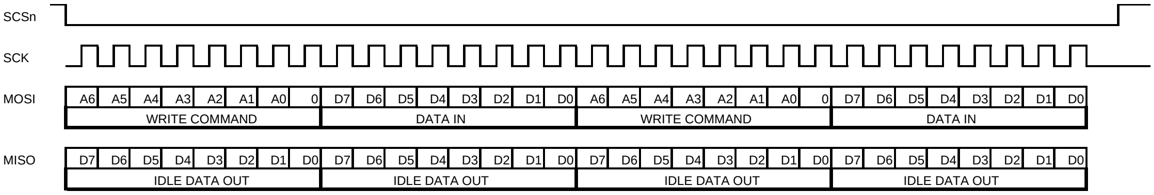

SCSn
SCK
MOSI
A6 A5 A4 A3 A2 A1 A0 0 D7 D6 D5 D4 D3 D2 D1 D0 A6 A5 A4 A3 A2 A1 A0 0 D7 D6 D5 D4 D3 D2 D1 D0
WRITE COMMAND
DATA IN
WRITE COMMAND
DATA IN
MISO
D7 D6 D5 D4 D3 D2 D1 D0 D7 D6 D5 D4 D3 D2 D1 D0 D7 D6 D5 D4 D3 D2 D1 D0 D7 D6 D5 D4 D3 D2 D1 D0
IDLE DATA OUT
IDLE DATA OUT
IDLE DATA OUT
IDLE DATA OUT

Figure 3. SPI Multiple Writes

Register Read Procedure

The ATW300 family of sensors support two different protocols for reading a register. These two read protocols are called “Persistent Read-Command” and “Non-Persistent Read-Command.” Upon reset the ATW300 defaults to the Persistent Read-Command protocol. The Non-Persistent Read-Command protocol (compatible with the ATW2xx family of sensors) can be selected by setting the SPI_PCL bit (IOC_REG[4]). The following sections describe the differences between the two read-command protocols.

SPI_PCL Bit (IOC_REG[4])	Selected SPI Read Protocol
0 (Default upon reset)	Persistent Read-Command
1	Non-Persistent Read-Command

Persistent Read-Command Protocol

In the Persistent Read-Command protocol, the slave accepts only the first read command after the assertion of SCSn. After receiving the first read command, the slave ignores MOSI as long as SCSn remains asserted. The register address specified in the first read command “persists” until the de-assertion of SCSn. Following the read command, the data transfer transitions into the Data Phase and the slave transmits the contents of the register specified in the read command. The Data Phase is repeated until SCSn is de-asserted.

Figure 4 shows a single-byte read, a method of reading a register one time. To read a register, the SPI master first drives SCSn low, sends the read command during the Command Phase (Time Slot 0), receives one data byte during the Data Phase (Time Slot 1), and then drives SCSn high. The master must issue a separate single-byte read for each separate register access.

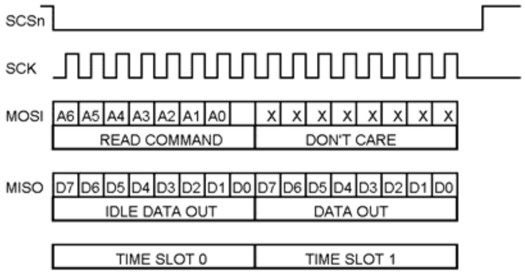


Figure 4. Single-Byte Read (Persistent Read-Command Protocol)

Figure 5 shows a multiple-byte read, a method of reading a register multiple times. During the Command Phase (Time Slot 0), the master sends a read command, specifying the address to be read. After the Command Phase, the data transfer remains in the Data Phase (Time Slots 1 through 4) until SCSn is de-asserted. For each time slot in the Data Phase, the slave reads the same register and transmits the data back to the master.

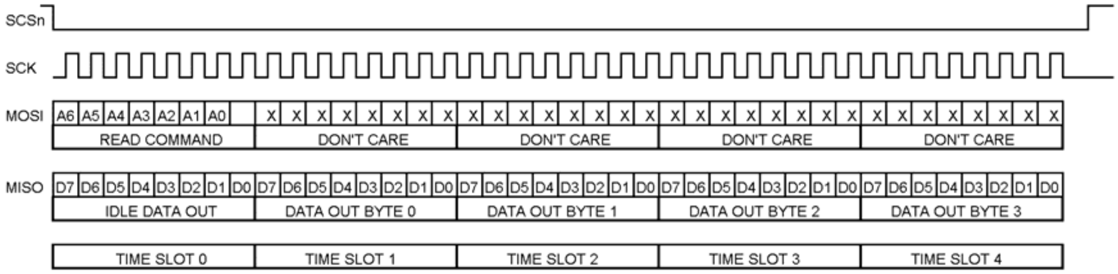


Figure 5. Multiple-Byte Read (Persistent Read-Command Protocol)

Figure 6 shows an example of the two separate read commands separated by the de-assertion of SCSn. In Time Slot 0, the master sends the first read command. In Time Slots 1 and 2, the slave reads the register and transmits the data back to the master. After Time Slot 2, the master drives SCSn high to terminate the data transmission. Then the master drives SCSn low and sends the second read command in Time slot 3. The slave reads the register and sends the data during Time Slot 4. After Time Slot 4, the master drives SCSn high to terminate the data transmission.

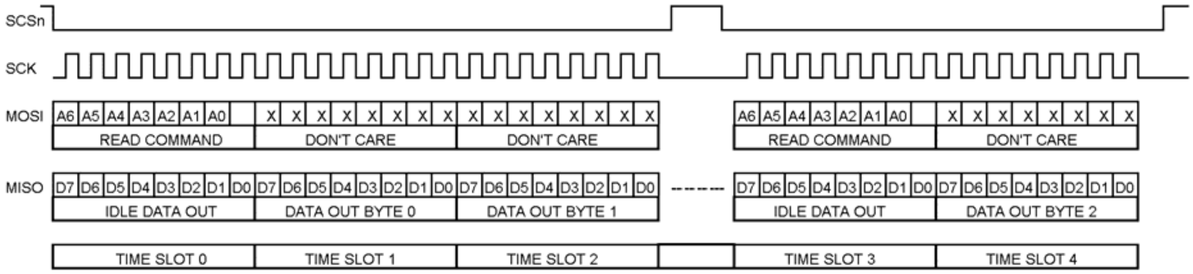


Figure 6. Multiple-Byte Read Separated by SCSn High (Persistent Read-Command Protocol)

Figure 7 shows an example of the SPI master pausing the data transfer by stopping SCK after Time Slot 2. SCSn remains asserted during the entire data transfer. The master resumes the data transfer by restarting SCK at the beginning of Time Slot 3.

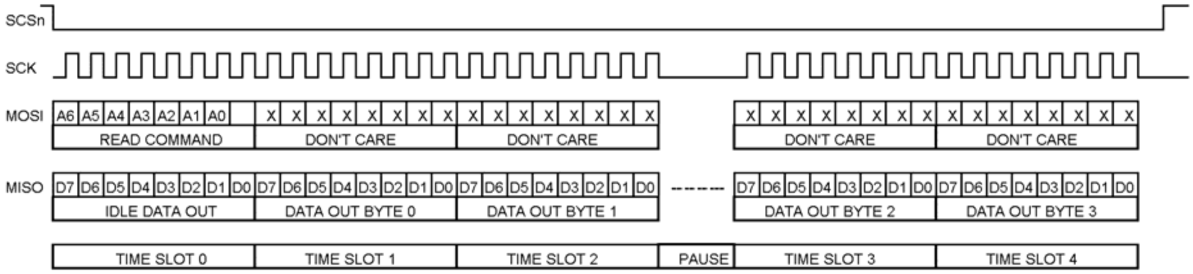


Figure 7. Multiple-Byte Read with SCK controlled Flow Control

(Persistent Read-Command Protocol)

Non-Persistent Read-Command Protocol

In the Non-Persistent Read-Command Protocol, the slave monitors MOSI for a read command as the slave transmits the requested data back to the master. The master must send a read command to the slave for each time slot that master is clocking data from the slave. Unlike the Persistent Read-Command Protocol, the Non-Persistent Read-Command Protocol does not require the host to de-assert SCSn before transmitting a read command. The master is allowed to read multiple registers while keeping SCSn asserted.

Figure 8 shows a single-byte read. To read a register, the master drives SCSn low, transmits the read command to the slave, receives one data byte from the slave, and then drives SCSn high. The master sends a second read command while the slave is sending the data. The slave will transmit the data requested by the second read command during the next read or write command.

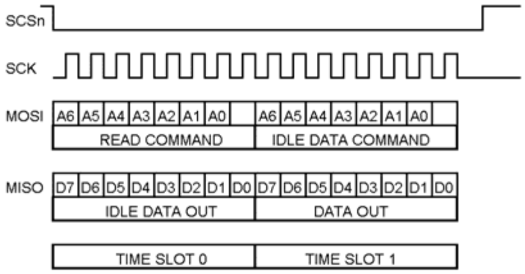


Figure 8. Single-Byte Read (Non-Persistent Read-Command Protocol)

Figure 9 shows a multiple-byte read. During Time Slot 0, the master sends a read command. During Time Slot 1, the slave sends the data for the register specified in Time Slot 0. In Time Slot 1 the master sends another read command. In Time Slot 2, the slave sends the data that was requested by the read command in Time Slot 1.

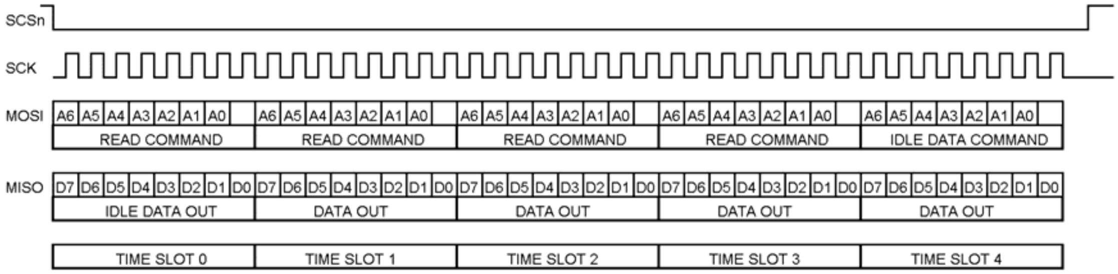


Figure 9. Multiple-Byte Read (Non-Persistent Read-Command Protocol)

The read command can be used to read the same register multiple times or to read multiple registers while SCSn is low. For example, in Figure 8, if identical read commands are issued in time slots 0 through 3, the same register will be read in time slots 1 through 4. However if four different register addresses are issued in time slots 0 through 3, then data from four different registers would be returned during time slots 1 through 4.

When using the Non-Persistent Read-Command Protocol to read a frame, ensure that the last read command of the frame is to read STA_REG instead of DAT_REG to avoid reading the first byte of the next frame.

Figure 10 shows an example of the SPI master using SCSn to pause and resume the data transfer. After time slot 2, the master drives SCSn high to pause data transmission. The data transfer pauses even though SCK continues to toggle. Then the master drives SCSn low and slave resumes transmission in Time Slot 3.

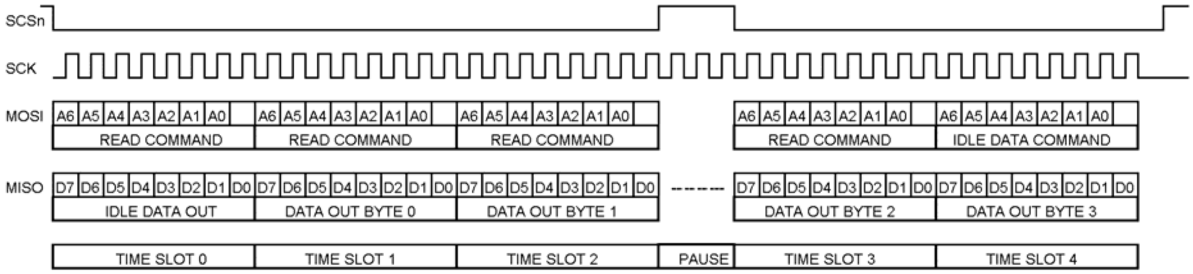


Figure 10. Multiple-Byte Read with SCSn controlled Flow Control

(Non-Persistent Read-Command Protocol)

Figure 11 shows an example of the master pausing the data transfer by stopping SCK after Time Slot 2. SCSn remains asserted during the entire data transfer. The master resumes the data transfer by restarting SCK at the beginning of Time Slot 3.

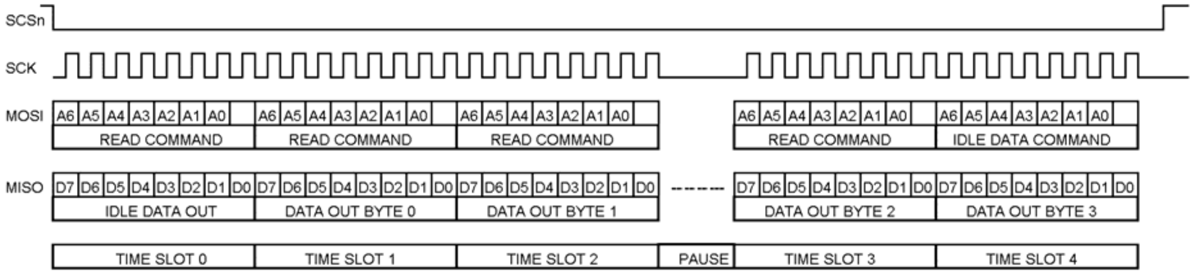


Figure 4. Multiple-Byte Read with SCK controlled Flow Control

(Non-Persistent Read-Command Protocol)

Additional Information

Please consult the "ATW300 Family Datasheet" for more information on bus and serial interface signals, timing, and other parameters.

Register Description

Tables 3 lists the user accessible registers of ATW300 family of sensors. Subsequent tables detail each of the internal user-accessible registers.

Table 3. ATW300 User Accessible Register Map

A0	*[IDX_REG]*	*Function*	*Acronym*
L	X	Index	IDX_REG
H	0x00	Command	CMD_REG
H	0x01	Data	DAT_REG
H	0x02	Status	STA_REG
H	0x03	Interrupt	INT_REG
H	0x04	Frame Interval Timer Control	FCON_REG
H	0x05	Auxiliary	AUX_REG
H	0x06	Chip ID 0	CID0_REG
H	0x07	Chip ID 1	CID1_REG
H	0x08	Sensor Test	OPS_REG
H	0x09	Buffer Control	BUFC_REG
H	0x0A	Clock Control 1	CLK0_REG
H	0x0B	RESERVED	
H	0x0C	RESERVED	
H	0x0D	RESERVED	
H	0x0E	RESERVED	
H	0x0F	RESERVED	
H	0x10	RESERVED	
H	0x11	RESERVED	
H	0x12	RESERVED	
H	0x13	RESERVED	
H	0x14	I/O Drive Strength Configuration	IOC_REG
H	0x15	Bulk Capacitance Detection Configuration	BCC_REG
H	0x16	Interrupt Enable	IEN_REG
H	0x17	RESERVED	
H	0x18	Threshold Crossing High and Low	THR_REG
H	0x19	Lower Mean Interrupt Threshold	LMT_REG
H	0x1A	Upper Variance Interrupt Threshold	UVT_REG
H	0x1B	Upper Threshold Crossing Count Interrupt	UCT_REG
H	0x1C	RESERVED	
H	0x1D	RESERVED	
H	0x1E	RESERVED	
H	0x1F	RESERVED	
H	0x20	Timestamp, Buffer 0, LSB	F0T0_REG
H	0x21	Timestamp, Buffer 0, MSB	F0T1_REG
H	0x22	Timestamp, Buffer 1, LSB	F1T0_REG
H	0x23	Timestamp, Buffer 1, MSB	F1T1_REG
H	0x24	Mean, right region, buffer 0	MRF0_REG

A0	*[IDX_REG]*	*Function*	*Acronym*
H	0x25	Mean, center region, buffer 0	MCF0_REG
H	0x26	Mean, left region, buffer 0	MLF0_REG
H	0x27	Mean, right region, buffer 1	MRF1_REG
H	0x28	Mean, center region, buffer 1	MCF1_REG
H	0x29	Mean, left region, buffer 1	MLF1_REG
H	0x2A	Mean, right region, least significant nibble, buffer 0	MRXF0_REG
H	0x2B	Mean, center region, least significant nibble, buffer 0	MCXF0_REG
H	0x2C	Mean, left region, least significant nibble, buffer 0	MLXF0_REG
H	0x2D	Mean, right region, least significant nibble, buffer 1	MRXF1_REG
H	0x2E	Mean, center region, least significant nibble, buffer 1	MCXF1_REG
H	0x2F	Mean, left region, least significant nibble, buffer 1	MLXF1_REG
H	0x30	Variance, right region	VRT_REG
H	0x31	Variance, center region	VCN_REG
H	0x32	Variance, left region	VLF_REG
H	0x33	Threshold crossing count, right region	TRT_REG
H	0x34	Threshold crossing count, center region	TCN_REG
H	0x35	Threshold crossing count, left region	TLF_REG
H	0x36	AGC, buffer 0	GSKF0_REG
H	0x37	AGC, buffer 1	GSKF1_REG
H	0x38	AGC, current	GSKC_REG
H	0x39	RESERVED	
H	0x3A	RESERVED	
H	0x3B	RESERVED	
H	0x3C	RESERVED	

Index Register (IDX_REG)

Access Conditions	A0 = 0
Access Type	Read/Write

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	REG_INDEX							
Reset Value	0	0	0	0	0	0	0	0

Bit	*Function*	*Description*
7:0	REG_INDEX	Register Index. This value is used as an index to access the data, control, and status registers. Set the index address of the register to be accessed into this register before reading from or writing to that register.

Command Register (CMD_REG)

Access Conditions	A0 = 1; [IDX_REG] = 0x00
Access Type	Read/Write

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	ADDR_RESET	MEASURE	FIND_GRAY	AUTO_RUN_EN	FRAME_TIMER_EN	SURF_CONTX_EN	MCLK_EN	RESERVED
Reset Value	0	0	0	0	0	0	0	0

Bit	*Function*	*Description*
7	ADDR_RESET	Reset frame buffer addresses. 0 = Data buffer address reset disabled. Must be negated by the host processor to allow reading and writing the buffer. 1 = Resets the read and write pointers for the data buffer. Also clears DATA_RDY and STA_REG[3:0].
6	MEASURE	Start measure operation. This performs an analog-to-digital conversion of the sensor array. 0 = Measure operation disabled. 1 = The transition of this bit from 0 to 1 starts the measure operation.
5	FIND_GRAY	Start gray level seek operation. 0 = Gray level seek operation is disabled. 1 = The transition of this bit from 0 to 1 enables a gray level seek operation of the current surface.
4	AUTO_RUN_EN	Auto-run enable 0 = Disables automatic run operation. 1 = If CMD_REG[2] is also 1, enables automatic run operation when a surface contact is detected. This invokes a gray level seek cycle followed by a measurement cycle. If bit[3] = 1, measurement cycles repeat automatically.
3	FRAME_TIMER_EN	Frame interval timer enable. 0 = Disables the frame interval timer. 1 = Enables the frame interval timer. The frame timer interval is programmed by the FRAME_DIV[3:0] bits. To enable the frame interval timer feature, set FRAME_TIMER_EN at the same time as the MEASURE bit.
2	SURF_CONTX_EN	Surface contact detector enable 0 = Disables the surface contact detection function and puts the sensor into the power-down mode. 1 = Enables the surface contact detection function,

		puts the sensor into the standby mode, and enables operation of the surface contact detect interrupt. If CMD_REG[4], AUTO_RUN_EN = 1, detection of a surface contact will automatically begin a calibration and measurement cycle.
1	MCLK_EN	Master clock enable 0 = Turns the internal clock off 1 = Starts the internal clock. This clock must be enabled for the sensor to function.
0	RESERVED	Reserved.

CMD_REG_[7:0]	Operating Mode
0x22	FindGray Operation.
0x42	Measure Operation.
0x62	Atomic FindGray-Measure Operation
0x4A	Frame Timer Mode.
0x04	Basic Finger Detect Mode
0x14	AutoRun Mode.

FindGray operation. The MCLK_EN bit must be written to ‘1’ before setting the FIND_GRAY bit. A ‘0’ to ‘1’ transition of the FIND_GRAY bit initiates a single FindGray operation. Do a FindGray operation before the initial Measure operation to calibrate the sensor. Optionally, do periodic FindGray operations as the finger traverses the sensor surface.

Measure operation. The MCLK_EN bit must be written to ‘1’ before setting the MEASURE bit. A ‘0’ to ‘1’ transition of the MEASURE bit initiates a single Measure operation. The Measure operation performs an A/D conversion on the sensor array and stores the results into one frame of the frame buffer. The MEASURE bit must be cleared and set for each sensor array conversion (unless the Frame Interval Timer is enabled).

Atomic FindGray-Measure operation. If the FIND_GRAY, MEASURE, and MCLK_EN bits are set at the same time, the device will do a FindGray operation followed by a Measure operation.

Frame Timer mode. It uses the Frame Interval Timer feature. The Frame Interval Timer feature allows image frames to be captured at a regular interval. The Frame Interval Timer is enabled if the FRAME_TIMER_EN bit is set concurrently with the MEASURE bit. When the frame timer is enabled, the sensor will automatically initiate measure operations at the programmed frame-timer interval as long as there is at least one empty frame buffer. If both frame buffers are full at the frame-timer interval, then the sensor will wait until the next frame-timer interval to attempt the Measure operation. As long as both buffers are full, the sensor will continue postponing the Measure operation by the frame-timer interval. The FRAME_DIV_[3:0] bits of the FCON register set the periodicity of the frame interval timer.

Basic Finger Detect Mode. It uses the Surface Contact Detector feature.

Surface Contact Detector feature. The Surface Contact Detector feature can be used to generate an interrupt when an increase in bulk capacitance is detected. Set the SURF_CONTX_EN bit to enable the Surface Contact Detector feature. The Surface Contact Detector is implemented as a Bulk Capacitance Detector. When the Surface Contact Detect feature is enabled, the sensor sleeps in a low power state and periodically wakes up and momentarily activates the Bulk Capacitance Detector. The period that the sensor sleeps is called the Finger Detect Interval and its periodicity is programmed by the DETECT_ON_[1:0] of the BUFC register. The Bulk Capacitance Detector triggers if the bulk capacitance exceeds a programmed threshold. The threshold or sensitivity of the Bulk Capacitance Detector can be adjusted by the BULK_CAP_[4:0] bits of the BCC register. If the Bulk Capacitance Detector triggers, the BULK_INT_STAT bit in the INT register will set. If the AutoRun feature was also enabled, then the sensor automatically initiates a Find Gray operation followed by a Measure operation.

AutoRun feature. The AutoRun feature can only be used when Advanced Finger Detect mode (Surface Contact Detector Enabled with comparator for the Mean, Variance and Threshold crossing) is used. When the AutoRun feature is enabled, the sensor waits in a low power state and periodically activates the Surface Contact Detector (Bulk Capacitance Detector). If the Bulk Capacitance Detector triggers, the sensor automatically initiates a FindGray operation followed by a Measure operation.

Data Register (DAT_REG)

Access Conditions	A0 = 1; [IDX_REG] = 0x01
Access Type	Read Only

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	DATA_ODD[3:0]				DATA_EVEN[3:0]			
Reset Value	Not reset							

Bit	*Function*	*Description*
7:4	DATA_ODD[3:0]	Frame buffer data for odd columns (1, 3, 5, … , 121, 123)
3:0	DATA_EVEN[3:0]	Frame buffer data for even columns (0, 2, 4, …, 120, 122)

Note: the 4-bit sensor data from each of two adjacent columns in one row is read from the data buffer as one byte. Thus, the data for the 124 columns is read in 62 bytes. A read of the data register increments the data buffer read pointer on the trailing (rising) edge of RDn.
Note: the frame buffers are not cleared on reset.

Status Register (STA_REG)

Access Conditions	A0 = 1; [IDX_REG] = 0x02
Access Type	Read Only

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	RESERVED						BUF_FULL_[1:0]	
Reset Value	0x00						0x00	

Bit	*Function*	*Description*
7:2	RESERVED	Reserved.
1	BUF_FULL_1	Frame buffer 1 full. A ‘1’ indicates that a complete frame has been stored in the buffer. The bit remains set until the data for the frame is completely read out and then resets to ‘0’ automatically.
0	BUF_FULL_0	Frame buffer 0 full. A ‘1’ indicates that a complete frame has been stored in the buffer. The bit remains set until the data for the frame is completely read out and then resets to ‘0’ automatically.

Interrupt Status Register (INT_STAT_REG)

Access Conditions	A0 = 1; [IDX_REG] = 0x03
Access Type	Read/Write

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	RESERVED			TC_IRQ_STAT	VAR_IRQ_STAT	MEAN_IRQ_STAT	BULK_IRQ_STAT	FR_RDY_IRA_STAT
Reset Value	0	0	0	0	0	0	0	0

Bit	*Function*	*Description*
7:5	RESERVED	Reserved.
4	TC_IRQ_STAT	Threshold-crossing interrupt flag.
3	VAR_IRQ_STAT	Variance interrupt flag.
2	MEAN_IRQ_STAT	Mean interrupt flag.
1	BULK_IRQ_STAT	Surface contact interrupt flag.
0	FR_RDY_IRQ_STAT	Frame ready interrupt flag.

Note: These interrupt status bits do not clear when read. An atomic read/clear would be needed to eliminate possible race conditions.

Frame Interval Timer Control (FCON_REG)

Access Conditions	A[0] = 1; [IDX_REG] = 0x04
Access Type	Read/Write

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	WRITE_ERR	MEASURE_ERR	MISS_FRAME_ERR	READ_EMPTY_ERR	FRAME_DIV_[3:0]			
Reset Value	0	0	0	0	1	1	1	1

Bit	*Function*	*Description*
7	WRITE_ERR	Buffer Write Error. 0 = No error. 1 = Indicates that an attempt was made to write into the frame buffer when it is full. Write a '0' to clear this bit.
6	MEASURE_ERR	Measure Error. 0 = No error. 1 = Indicates that an attempt was made to initiate a measure operation when the frame buffer is full. Write a '0' to clear this bit.
5	MISS_FRAME_ERR	Missed Frame Error. 0 = No error. 1 = Indicates than an internal operation was completed that required writing data into the frame buffer but the buffer was full. Write a '0' to clear this bit.
4	READ_EMPTY_ERR	Read Empty Buffer Error. 0 = No error. 1 = Indicates than an attempt was made to read from the frame buffer when it is empty. Write a '0' to clear this bit.
3:0	FRAME_DIV_[3:0]	Frame Divider. FRAME_DIV[3:0] sets the frame timer interval, which is the delay between automatic Measure operations. This value is loaded into the internal frame interval timer when FRAME_TIM_EN, CMD_REG[3], is written to '0'.

FRAME_DIV[3:0] sets the frame timer interval from the last Measure operation to the start of the next Measure operation. The smallest frame timer interval unit is t_MEASURE (see CLK0 description). The value of FRAME_DIV[3:0] selects the number of interval units to delay before the start of the next Measure operation. For example, for MCLK_F2X = 1 and FRAME_DIV[3:0] = 3, there will be nominally 3*256 us between the last Measure operation and the start of the next Measure operation. The table below shows the time spent during Measure (actual A/D conversion) and the Delay between Measure operations. The table assumes that a buffer was emptied some time between Measure 1 and 2.

Measure 0	Delay	Measure 1	Delay	Measure 3
256 us	3*256 us	256 us	3*256 us	256 us

The frame rate can be computed with the following equation:

Frame Rate = 1/[((1 + FRAME_DIV[3:0]) * t_MEASURE) /frame]

The following table shows the computed frame rate for a given FRAME_DIV[3:0] and MCLK_F2X setting.

FRAME_DIV[3:0]	**Nominal Frame Rate (fps)**	**Nominal Frame Rate (fps)**
	MCLK_F2X = 0	**MCLK_F2X = 1**
0	Not Valid	Not Valid
1	977	1953
2	651	1302
3	488	977
4	391	781
5	326	651
6	279	558
7	244	488
8	217	434
9	195	391
A	178	355
B	163	326
C	150	300
D	140	279
E	130	260
F	122	244

FRAME_DIV[3:0] = 0 is not a valid setting. The frame timer won’t initiate Measure operations if FRAME_DIV[3:0] = 0.

Auxiliary Register (AUX_REG)

Access Conditions	A0 = 1; [IDX_REG] = 0x05
Access Type	Read Only

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	RESERVED			BUSY	RESERVED			
Reset Value	Not reset			0	Not reset			

Bit	*Function*	*Description*
7:5	RESERVED	Reserved.
4	BUSY	Busy. 0 = A measure or find gray operation is not in progress. 1 = A measure or find gray operation is in progress.
3:0	RESERVED	Reserved.

Chip ID 0 Register (CID0_REG)

Access Conditions	A0 = 1, [IDX_REG] = 0x06
Access Type	Read Only

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	CHIP_ID_ROM0 [7:0]							
Reset Value (REV AA)	0	0	0	1	0	0	1	0

Bit	*Function*	*Description*
7:0	CHIP_ID_ROM0 [7:0]	Chip ID ROM 0, bits [7:0]

Chip ID 1 Register (CID1_REG)

Access Conditions	A0 = 1, [IDX_REG] = 0x07
Access Type	Read Only

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	CHIP_ID_ROM1 [7:0]							
Reset Value (REV AA)	0	0	1	1	0	0	0	0

Bit	*Function*	*Description*
7:0	CHIP_ID_ROM1 [7:0]	Chip ID ROM 1, bits [7:0]

Sensor Test Register (OPS_REG)

Access Conditions	A0 = 1; [IDX_REG] = 0x08
Access Type	Read/Write

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	RESERVED				INTEG_CAP_3	INTEG_CAP_2	INTEG_CAP_1	INTEG_CAP_0
Reset Value	0	0	0	0	1	1	1	1

Bit	*Function*	*Description*
7:4	RESERVED	Reserved. These bits should be set to 0
3	INTEG_CAP_3	Select integration capacitor 3. 0 = Integration capacitor 3 is not selected. 1 = Integration capacitor 3 is selected.
2	INTEG_CAP_2	Select integration capacitor 2. 0 = Integration capacitor 2 is not selected. 1 = Integration capacitor 2 is selected.
1	INTEG_CAP_1	Select integration capacitor 1. 0 = Integration capacitor 1 is not selected. 1 = Integration capacitor 1 is selected.
0	INTEG_CAP_0	Select integration capacitor 0. 0 = Integration capacitor 0 is not selected. 1 = Integration capacitor 0 is selected.

Notes:

1. A value of OPS_REG[3:0] = 0b0011, which selects capacitors 0 and 1, is recommended for ideal sensitivity.

Buffer Control Register (BUFC_REG)

Access Conditions	A0 = 1; [IDX_REG] = 0x09
Access Type	Read/Write

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	TR_EN	DETECT_ON[1:0]		RESERVED				FORCE _RESE T
Reset Value	0	0	0	0	0	0	0	0

Bit	*Function*	*Description*
7	TR_EN	Trailer Enable. 0 = Do not append the trailer to the frame data. 1 = Append the trailer to the frame data.
6:5	DETECT_ON	Nominal Finger Detect Interval. Programs the surface contact detect rate as follows: 00 = Detect every ~ 0.05 s 01 = Detect every ~ 0.10 s 10 = Detect every ~ 0.20 s 11 = Detect every ~ 0.40 s The Finger Detect Interval is based on the internal SCLK and has no relationship to MCLK or the MCLK_F2X bit.
4:1	RESERVED	Reserved.
0	FORCE_RESET	Force reset. 0 = The force reset function is not active. 1 = A 0-to-1 transition of this bit replicates the power-on reset function. The bit is self clearing and does not need to be reset.

Clock Control Register (CLK0_REG)

Access Conditions	A0 = 1; [IDX_REG] = 0x0A
Access Type	Read/Write

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	AUTO _VAR _EN	SCLK _EN	MCLK _F2X	RESERVED				
Reset Value	0	0	0	0	0	0	0	0

Bit	*Function*	*Description*
7	AUTO_VAR_EN	Auto Variance Calculation Enable. 0 = Don’t automatically calculate the variance and threshold crossings in Auto-Run mode. 1 = Automatically calculate the variance and threshold crossings after the completion of the MEASURE operation in Auto-Run mode.
6	SCLK_EN	SCLK Enable. 0 = Disable the SCLK. 1 = Enable the SCLK for Timestamping.
5	MCLK_F2X	Main clock rate control 0 = The sensor main internal clock operates at normal speed (~8 MHz) 1 = The sensor main internal clock operates at approximately 2x normal speed (~16 MHz)
4:0	RESERVED	Reserved.

I/O Drive Strength Configuration Register (IOC_REG)

Access Conditions	A0 = 1; [IDX_REG] = 0x14
Access Type	Read/Write

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	F_PWR_DN	P_PWR_DN	RESERVED	SPI_PCL	RESERVED		DRV_EN_[1:0]	
Reset Value	0	0	0	0	0	0	1	1

Bit	*Function*	*Description*
7	F_PWR_DN	Full Power Down 0 = Sensor operational 1 = Full Power Down mode
6	P_PWR_DN	Partial Power Down 0 = Normal active mode of operation 1 = Partial Power Down mode IMPORTANT: When this bit is set (sensor in partial power down mode), sensor registers cannot be read, only certain registers can be written. In order to read any status register, this bit must be set to 0.
5	RESERVED	Reserved
4	SPI_PCL	Select SPI Protocol. The SPI_PCL bit selects between the Persistent Read-Command protocol and the Non-Persistent Read-Command protocol 0 = Persistent Read-Command Protocol. The SPI Read Command transmitted at the beginning of the SPI transfer remains in effect until SCS- is de-asserted after which The SPI Read Command is “forgotten.” The MOSI bits after the first Read Command are “don’t cares” while SCS- is asserted. 1 = Non-Persistent Read-Command Protocol. An SPI Read Command must be transmitted before each byte to be read.
3:2	RESERVED	Reserved.
1:0	DRV_EN_[1:0]	Drive Strength Enable. Configures the output drive strength of the D[7:0] pins. 00 = Minimum drive strength. 01 = Medium drive strength. Suggested for 2.5V VDDO. 10 = Same as 01 setting. 11 = Maximum drive strength. Suggested for 1.8V VDDO.

Bulk Capacitance Detection Configuration Register (BCC_REG)

Access Conditions	A0 = 1; [IDX_REG] = 0x15
Access Type	Read/Write

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	RESERVED	BULK_CMP[1:0]		BULK_CAP[4:0]				
Reset Value	0	0	0	0	0	0	0	0

Bit	*Function*	*Description*
7	RESERVED	Reserved.
6:5	BULK_CMP[1:0]	Comparator hysteresis. Provides finer adjustment to the base sensitivity setting.
4:0	BULK_CAP[4:0]	Bulk reference capacitor. Increasing values of BULK_CAP[4:0] decreases the surface contact detector sensitivity. 0b00000 = Most sensitive setting 0b11111 = Least sensitive setting Note: The least sensitive setting would cause the surface contact detector to self trigger even without a finger touching the sensor. The least sensitive setting may not trigger some fingers.

Interrupt Enable Register (IEN_REG)

Access Conditions	A0 = 1; [IDX_REG] = 0x16
Access Type	Read/Write

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	RESERVED	FD_IRQ_EN	SEL_AND	TC_IRQ_EN	VAR_IRQ_EN	MEAN_IRQ_EN	BULK_IRQ_EN	FR_RDY_IRQ_EN
Reset Value	0	0	0	0	0	0	0	0

Bit	*Function*	*Description*
7	RESERVED	Reserved.
6	FD_IRQ_EN	Enable finger detect interrupt
5	SEL_AND	Select the "AND" of the interrupt sources 0 = Interrupt sources are OR'd to generate interrupt 1 = Interrupt sources are AND'd to generate interrupt
4	TC_IRQ_EN	Enable threshold-crossing interrupt
3	VAR_IRQ_EN	Enable variance interrupt
2	MEAN_IRQ_EN	Enable mean interrupt
1	BULK_IRQ_EN	Enable contact interrupt
0	FR_RDY_IRQ_EN	Enable frame ready interrupt

Threshold Crossing High and Low Register (THR_REG)

Access Conditions	A[0] = 1; [IDX_REG] = 0x18
Access Type	Read/Write

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	THR_UPPER_[3:0]				THR_LOWER_[3:0]			
Reset Value	0	0	0	0	0	0	0	0

Bit	*Function*	*Description*
7:4	THR_UPPER_[3:0]	Upper threshold.
3:0	THR_LOWER_[3:0]	Lower threshold.

Lower Mean Interrupt Threshold Register (LMT_REG)

Access Conditions	A[0] = 1; [IDX_REG] = 0x19
Access Type	Read/Write

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	LMT_[7:0]							
Reset Value	0	0	0	0	0	0	0	0

Bit	*Function*	*Description*
7:0	LMT_[7:0]	Lower mean interrupt threshold. The MEAN_IRQ flag will set after a bulk finger detect event if at least one of the means from the three regions (right, center, or left) meets or falls below value in the LMT register.

Upper Variance Interrupt Threshold (UVT_REG)

Access Conditions	A[0] = 1; [IDX_REG] = 0x1A
Access Type	Read/Write

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	UVT_[7:0]							
Reset Value	0	0	0	0	0	0	0	0

Bit	*Function*	*Description*
7:0	UVT_[7:0]	Upper variance interrupt threshold. The VAR_IRQ flag will set after a bulk finger detect event if at least one of the variances from the three regions (left, center, or right) exceeds the UVT.

Upper Threshold-crossing Count Interrupt Threshold (UCT_REG)

Access Conditions	A[0] = 1; [IDX_REG] = 0x1B
Access Type	Read/Write

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	RESERVED	UCT_[6:0]						
Reset Value	0	0	0	0	0	0	0	0

Bit	*Function*	*Description*
7	RESERVED	Reserved.
6:0	UCT_[6:0]	Upper threshold-crossing interrupt threshold. The TC_IRQ flag will set after a bulk finger detect event if at least one of the threshold-crossing counts from the three regions (left, center, or right) exceeds the UCT.

Timestamp, Buffer 0, LSB Register (F0T0_REG)

Access Conditions	A0 = 1; [IDX_REG] = 0x20
Access Type	Read Only

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	F0T0_[7:0]							
Reset Value	0	0	0	0	0	0	0	0

Bit	*Function*	*Description*
7:0	F0T0_[7:0]	LSB of timestamp for frame 0

Timestamp, Buffer 0, MSB Register (F0T1_REG)

Access Conditions	A0 = 1; [IDX_REG] = 0x21
Access Type	Read Only

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	F0T1_[15:8]							
Reset Value	0	0	0	0	0	0	0	0

Bit	*Function*	*Description*
15:8	F0T1_[15:8]	MSB of timestamp for frame 0

Timestamp, Buffer 1, LSB Register (F1T0_REG)

Access Conditions	A0 = 1; [IDX_REG] = 0x22
Access Type	Read Only

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	F1T0_[7:0]							
Reset Value	0	0	0	0	0	0	0	0

Bit	*Function*	*Description*
7:0	F1T0_[7:0]	LSB of timestamp for frame 1

Timestamp, Buffer 1, MSB Register (F1T1_REG)

Access Conditions	A0 = 1; [IDX_REG] = 0x23
Access Type	Read Only

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	F1T1_[15:8]							
Reset Value	0	0	0	0	0	0	0	0

Bit	*Function*	*Description*
7:0	F1T1_[15:8]	MSB of timestamp for frame 1

Mean, right region, buffer 0 (MRF0_REG)

Access Conditions	A[0] = 1; [IDX_REG] = 0x24
Access Type	Read Only

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	MRF0_[7:0]							
Reset Value	0	0	0	0	0	0	0	0

Bit	*Function*	*Description*
7:0	MRF0_[7:0]	Mean of right region of frame 0, in 4.4 format. The mean is undefined during the Measure operation, but is valid immediately after the Measure operation completes.

Note: For mean and variance, 60 of the 124 columns are used for the computation. The 60 columns include every even-numbered column except columns 0 and 122. The left region spans columns 2 through 60. The center region spans columns 32 through 90. The right region spans columns 62 through 120.

Mean, center region, buffer 0 (MCF0_REG)

Access Conditions	A[0] = 1; [IDX_REG] = 0x25
Access Type	Read Only

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	MCF0_[7:0]							
Reset Value	0	0	0	0	0	0	0	0

Bit	*Function*	*Description*
7:0	MCF0_[7:0]	Mean of center region of frame 0, in 4.4 format. The mean is undefined during the Measure operation, but is valid immediately after the Measure operation completes.

Mean, left region, buffer 0 (MRF0_REG)

Access Conditions	A[0] = 1; [IDX_REG] = 0x26
Access Type	Read Only

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	MLF0_[7:0]							
Reset Value	0	0	0	0	0	0	0	0

Bit	*Function*	*Description*
7:0	MLF0_[7:0]	Mean of left region of frame 0, in 4.4 format. The mean is undefined during the Measure operation, but is valid immediately after the Measure operation completes.

Mean, right region, buffer 1 (MLF1_REG)

Access Conditions	A[0] = 1; [IDX_REG] = 0x27
Access Type	Read Only

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	MRF1_[7:0]							
Reset Value	0	0	0	0	0	0	0	0

Bit	*Function*	*Description*
7:0	MRF1_[7:0]	Mean of right region of frame 1, in 4.4 format. The mean is undefined during the Measure operation, but is valid immediately after the Measure operation completes.

Mean, center region, buffer 1 (MCF1_REG)

Access Conditions	A[0] = 1; [IDX_REG] = 0x28
Access Type	Read Only

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	MCF1_[7:0]							
Reset Value	0	0	0	0	0	0	0	0

Bit	*Function*	*Description*
7:0	MCF1_[7:0]	Mean of center region of frame 1, in 4.4 format. The mean is undefined during the Measure operation, but is valid immediately after the Measure operation completes.

Mean, left region, buffer 1 (MLF1_REG)

Access Conditions	A[0] = 1; [IDX_REG] = 0x29
Access Type	Read Only

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	MLF1_[7:0]							
Reset Value	0	0	0	0	0	0	0	0

Bit	*Function*	*Description*
7:0	MLF1_[7:0]	Mean of left region of frame 1, in 4.4 format. The mean is undefined during the Measure operation, but is valid immediately after the Measure operation completes.

Mean, right region, least significant nibble, buffer 0 (MRXF0_REG)

Access Conditions	A[0] = 1; [IDX_REG] = 0x2A
Access Type	Read Only

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	RESERVED				MRXF0_[3:0]			
Reset Value	0	0	0	0	1	0	0	0

Bit	*Function*	*Description*
7:4	RESERVED	Reserved.
3:0	MRXF0_[3:0]	Least significant bits of mean of right region of frame 0.

Mean, center region, least significant nibble, buffer 0 (MCXF0_REG)

Access Conditions	A[0] = 1; [IDX_REG] = 0x2B
Access Type	Read Only

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	RESERVED				MCXF0_[3:0]			
Reset Value	0	0	0	0	1	0	0	0

Bit	*Function*	*Description*
7:4	RESERVED	Reserved.
3:0	MCXF0_[3:0]	Least significant bits of mean of center region of frame 0.

Mean, left region, least significant nibble, buffer 0 (MLXF0_REG)

Access Conditions	A[0] = 1; [IDX_REG] = 0x2C
Access Type	Read Only

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	RESERVED				MLXF0_[3:0]			
Reset Value	0	0	0	0	1	0	0	0

Bit	*Function*	*Description*
7:4	RESERVED	Reserved.
3:0	MLXF0_[3:0]	Least significant bits of mean of left region of frame 0.

Mean, right region, least significant nibble, buffer 1 (MRXF1_REG)

Access Conditions	A[0] = 1; [IDX_REG] = 0x2D
Access Type	Read Only

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	RESERVED				MRXF1_[3:0]			
Reset Value	0	0	0	0	1	0	0	0

Bit	*Function*	*Description*
7:4	RESERVED	Reserved.
3:0	MRXF1_[3:0]	Least significant bits of mean of right region of frame 1.

Mean, center region, least significant nibble, buffer 1 (MCXF1_REG)

Access Conditions	A[0] = 1; [IDX_REG] = 0x2E
Access Type	Read Only

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	RESERVED				MCXF1_[3:0]			
Reset Value	0	0	0	0	1	0	0	0

Bit	*Function*	*Description*
7:4	RESERVED	Reserved.
3:0	MCXF1_[3:0]	Least significant bits of mean of center region of frame 1.

Mean, left region, least significant nibble, buffer 1 (MLXF1_REG)

Access Conditions	A[0] = 1; [IDX_REG] = 0x2F
Access Type	Read Only

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	RESERVED				MLXF1_[3:0]			
Reset Value	0	0	0	0	1	0	0	0

Bit	*Function*	*Description*
7:4	RESERVED	Reserved.
3:0	MLXF1_[3:0]	Least significant bits of mean of right region of frame 1.

Variance, right region (VRT_REG)

Access Conditions	A[0] = 1; [IDX_REG] = 0x30
Access Type	Read Only

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	VRT_[7:0]							
Reset Value	0	0	0	0	0	0	0	0

Bit	*Function*	*Description*
7:0	VRT_[7:0]	Variance of right region of frame most recently output to host, in 4.4 format. The variance is undefined while reading a frame. The variance is valid after reading the last byte of the current frame and before reading the first byte of the next frame.

Variance, center region (VCN_REG)

Access Conditions	A[0] = 1; [IDX_REG] = 0x31
Access Type	Read Only

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	VCN_[7:0]							
Reset Value	0	0	0	0	0	0	0	0

Bit	*Function*	*Description*
7:0	VCN_[7:0]	Variance of center region of frame most recently output to host, in 4.4 format. The variance is undefined while reading a frame. The variance is valid after reading the last byte of the current frame and before reading the first byte of the next frame.

Variance, left region (VLF_REG)

Access Conditions	A[0] = 1; [IDX_REG] = 0x32
Access Type	Read Only

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	VLF_[7:0]							
Reset Value	0	0	0	0	0	0	0	0

Bit	*Function*	*Description*
7:0	VLF_[7:0]	Variance of left region of frame most recently output to host, in 4.4 format. The variance is undefined while reading a frame. The variance is valid after reading the last byte of the current frame and before reading the first byte of the next frame.

Threshold crossing count, right region (TRT_REG)

Access Conditions	A[0] = 1; [IDX_REG] = 0x33
Access Type	Read Only

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	TRT_[7:0]							
Reset Value	0	0	0	0	0	0	0	0

Bit	*Function*	*Description*
7:0	TRT_[7:0]	Threshold crossing count of right region of frame most recently output to host. The threshold-crossing count is undefined while reading a frame. The threshold-crossing is valid after reading the last byte of the current frame and before reading the first byte of the next frame.

Note: The right region spans from column 62 through column 123, inclusive.

Threshold crossing count, center region (TCN_REG)

Access Conditions	A[0] = 1; [IDX_REG] = 0x34
Access Type	Read Only

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	TCN_[7:0]							
Reset Value	0	0	0	0	0	0	0	0

Bit	*Function*	*Description*
7:0	TCN_[7:0]	Threshold crossing count of center region of frame most recently output to host. The threshold-crossing count is undefined while reading a frame. The threshold-crossing is valid after reading the last byte of the current frame and before reading the first byte of the next frame.

Note: The center region spans columns 32 through 93, inclusive.

Threshold crossing count, left region (TLF_REG)

Access Conditions	A[0] = 1; [IDX_REG] = 0x35
Access Type	Read Only

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	TLF_[7:0]							
Reset Value	0	0	0	0	0	0	0	0

Bit	*Function*	*Description*
7:0	TLF_[7:0]	Threshold crossing count of left region of frame most recently output to host. The threshold-crossing count is undefined while reading a frame. The threshold-crossing is valid after reading the last byte of the current frame and before reading the first byte of the next frame.

Note: The left region spans from column 0 through 61, inclusive.

AGC, Buffer 0 (GSKF0_REG)

Access Conditions	A[0] = 1; [IDX_REG] = 0x36
Access Type	Read/Write

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	RESERVED	AGCF0_[6:0]						
Reset Value	0	0	0	0	0	0	0	0

Bit	*Function*	*Description*
7	RESERVED	Reserved.
6:0	AGCF0_[6:0]	AGCF0_[6:0] is the saved FindGray value returned in the trailer for frame buffer 0. AGCF0_[6:0] is the FindGray value computed for frame buffer 0, but not necessarily the FindGray value used in the Measure operation for frame buffer 0.

Note: This register can be read in lieu of reading the FindGray value in the trailer.

The value in GSKF0_REG is the output of the FindGray circuit at the time of the Measure operation for buffer 0.

GSKF0_REG isn’t intended to reflect the current state of the FindGray circuit. Instead GSKF0_REG saves the FindGray state for the Measure operation for buffer 0. The value in GSKF0_REG won’t change until the next Measure operation for buffer 0, regardless of how many FindGray operations are executed. To get the current state of the FindGray circuit, read the GSKC_REG instead.

AGC, Buffer 1 (GSKF1_REG)

Access Conditions	A[0] = 1; [IDX_REG] = 0x37
Access Type	Read/Write

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	RESERVED	GSKF1_[6:0]						
Reset Value	0	0	0	0	0	0	0	0

Bit	*Function*	*Description*
7	RESERVED	Reserved.
6:0	GSKF1_[6:0]	GSKF1_[6:0] is the saved FindGray value returned in the trailer for frame buffer 1. GSKF1_[6:0] is the FindGray value computed for frame buffer 1, but not necessarily the FindGray value used in the Measure operation for frame buffer 1.

Note: This register can be read in lieu of reading the FindGray value in the trailer.

The value in GSKF1_REG is the output of the FindGray circuit at the time of the Measure operation for buffer 1.

GSKF1_REG isn’t intended to reflect the current state of the FindGray circuit. Instead GSKF1_REG saves the FindGray state for the Measure operation for buffer 1. The value in GSKF1_REG won’t change until the next Measure operation for buffer 1, regardless of how many FindGray operations are executed. To get the current state of the FindGray circuit, read the GSKC_REG instead.

AGC Current (GSKC_REG)

Access Conditions	A[0] = 1; [IDX_REG] = 0x38
Access Type	Read Only

Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Function	RESERVED	GSKC_[6:0]						
Reset Value	0	0	0	0	0	1	1	1

Bit	*Function*	*Description*
7	RESERVED	Reserved.
6:0	GSKC_[6:0]	Find Gray Result. GSKC_[6:0] is the current FindGray (recalibration) result.

ATW300 Measurement Data

A measurement image (‘frame’) from the Atrua sensor is either 496 bytes (3,968 bits) or 512 bytes (4,096 bits) long. The frame contains 4-bit gray values for each of the 992 sensing elements (8 rows, each with 124 columns), plus, if BUFC_REG_[7] (TR_EN) is set, a 14-byte trailer, including a 2-byte timestamp. If TR_EN is clear, then no trailer is sent.

Table 4. ATW300 Measurement Data Structure

Byte	*Bit*								*Description*	*Format*
	7	*6*	*5*	*4*	*3*	*2*	*1*	*0*		
0	O	O	O	O	E	E	E	E	Row 0, columns 0 and 1	E = 4-bit gray value, even pixel. O = 4-bit gray value, odd pixel.
1	O	O	O	O	E	E	E	E	Row 0, columns 2 and 3	
2	O	O	O	O	E	E	E	E	Row 0, columns 4 and 5	
…										
61	O	O	O	O	E	E	E	E	Row 0, columns 122 and 123	
62	O	O	O	O	E	E	E	E	Row 1, columns 0 and 1	
…										
495	O	O	O	O	E	E	E	E	Row 7, columns 122 and 123	
496	T	T	T	T	T	T	T	T	LSB of 16-bit timestamp	16-bit integer timestamp in 16.0 format, 100 microsecond resolution.
497	T	T	T	T	T	T	T	T	MSB of 16-bit timestamp	
498	I	I	I	I	F	F	F	F	Estimated mean of right region of frame	8-bit estimated means in 4.4 format; I = integer part, F = fractional part
499	I	I	I	I	F	F	F	F	Estimated mean of center region of	
500	I	I	I	I	F	F	F	F	Estimated mean of left region of frame	
501	X	X	X	X	F	F	F	F	Remainder of estimated mean of right region of frame	Least significant 4 bits of estimated means; X = not used, F = fractional part
502	X	X	X	X	F	F	F	F	Remainder of estimated mean of center region of	
503	X	X	X	X	F	F	F	F	Remainder of estimated mean of left region of frame	

Byte	*Bit*								*Description*	*Format*
	7	*6*	*5*	*4*	*3*	*2*	*1*	*0*		
504	I	I	I	I	F	F	F	F	Estimated variance of right region of frame	8-bit estimated variances in 4.4 format; I = integer part, F = fractional part
505	I	I	I	I	F	F	F	F	Estimated variance of center region of frame	
506	I	I	I	I	F	F	F	F	Estimated variance of left region of frame	
507	Z	Z	Z	Z	Z	Z	Z	Z	Zero-crossing count for right region of frame	8-bit zero crossing count in 8.0 format; Z = integer part
508	Z	Z	Z	Z	Z	Z	Z	Z	Zero-crossing count for center region of frame	
509	Z	Z	Z	Z	Z	Z	Z	Z	Zero-crossing count for left region of frame	
510	U	U	U	U	L	L	L	L	Threshold High and Low (THR_REG contents)	
511	X	I	I	I	I	I	I	I	AGC result of the frame (GSKFx_REG contents)	

Sample Operation Flowcharts

Figures 12 and 13 are sample flowcharts describing the operations required to perform an automatic gain control (AGC) operation and to digitize a single frame of data for the Atrua's sensor. Figure 14 is a flowchart describing a GetImage() function implemented in a driver for acquiring a frame from the sensor. This procedure includes the AGC and Digitize Frame operations in slightly modified form. The data acquired in this manner would next be passed onto the fingerprint reconstruction software running on the host processor or other device to construct the full fingerprint from the various frames of acquired data, and to extract the minutiae points necessary for the fingerprint authentication process.

The read and write operations normally require two cycles. The first cycle writes the Index register with the address of the target register. The second cycle performs the actual read or write transaction with the target register. The Index register is static, and will hold its contents until power is removed or the device is reset. When reading (or writing) the same register twice or more, the Index register need only be written once. For example, when reading a frame from the Data buffer, write 0x01 into IDX_REG once, and then read DAT_REG 496 times or 512 times, depending on the status of TR_EN (trailer enabled).

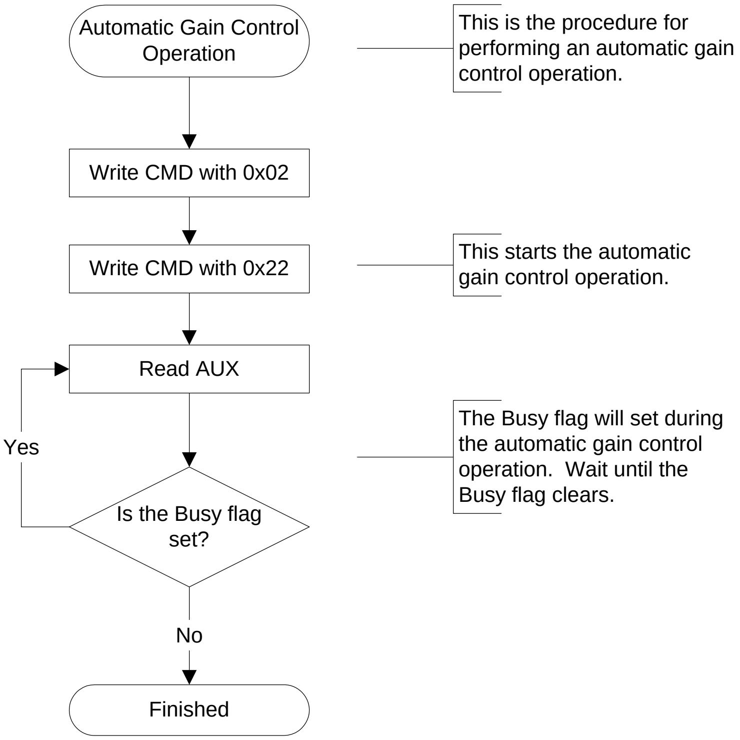
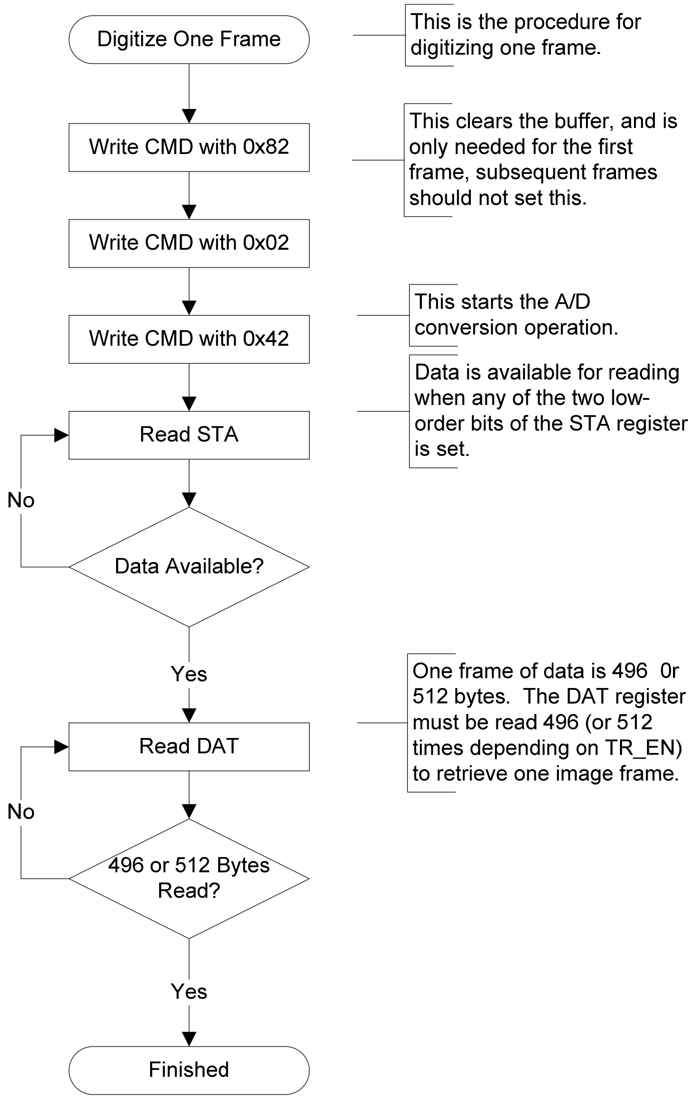

Automatic Gain Control Operation
Write CMD with 0x02
Write CMD with 0x22
Read AUX
Yes
Is the Busy flag set?
No
Finished
This is the procedure for performing an automatic gain control operation.
This starts the automatic gain control operation.
The Busy flag will set during the automatic gain control operation. Wait until the Busy flag clears.

Figure 12. Automatic Gain Control Operation Flowchart



This is the procedure for digitizing one frame.

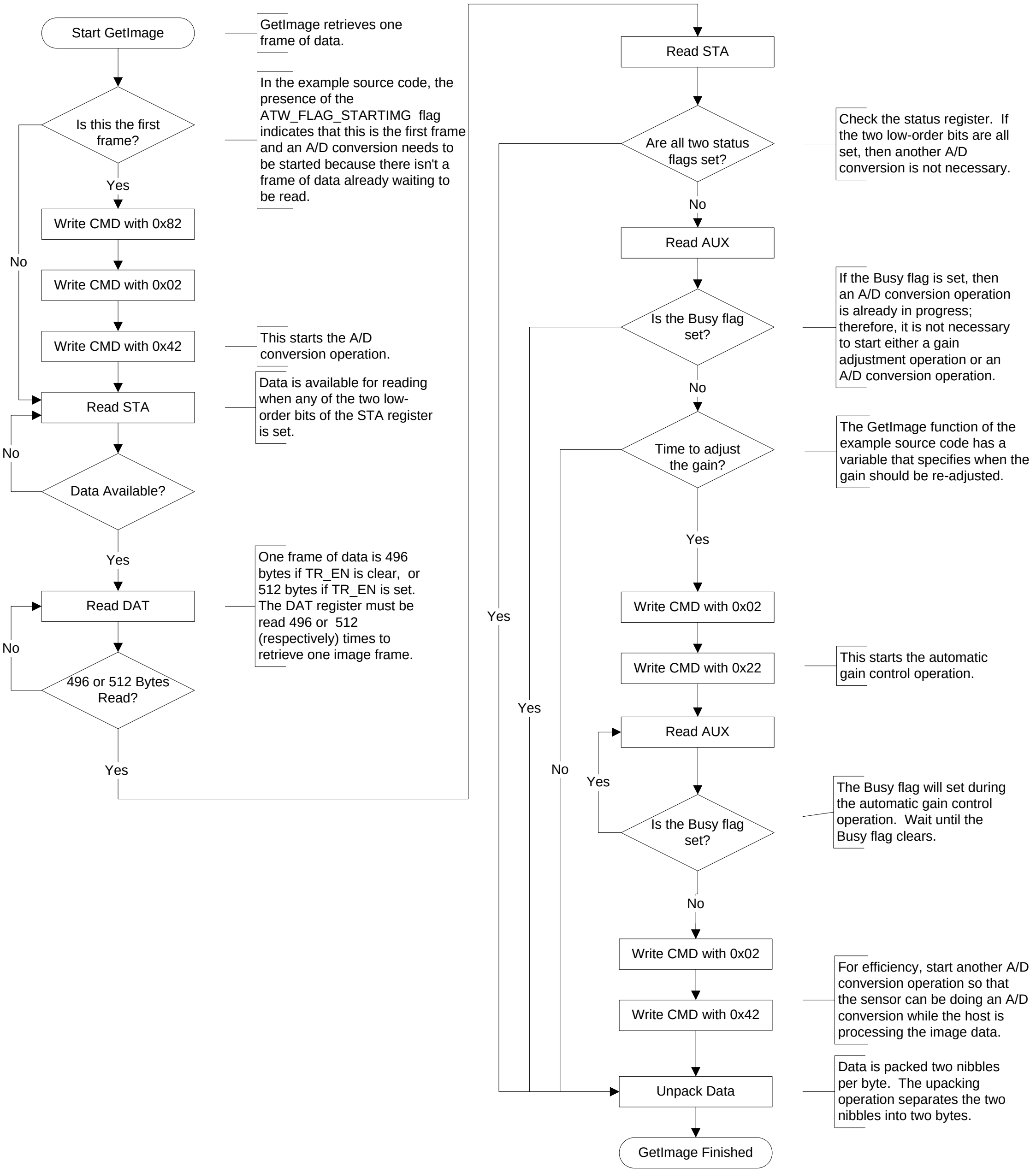
This clears the buffer, and is only needed for the first frame, subsequent frames should not set this.

This starts the A/D conversion operation.

Data is available for reading when any of the two low-order bits of the STA register is set.

One frame of data is 496 0r 512 bytes. The DAT register must be read 496 (or 512 times depending on TR_EN) to retrieve one image frame.

Figure 13. Measurement Operation Flowchart



GetImage retrieves one frame of data.

In the example source code, the presence of the ATW_FLAG_STARTIMG flag indicates that this is the first frame and an A/D conversion needs to be started because there isn't a frame of data already waiting to be read.

This starts the A/D conversion operation.

Data is available for reading when any of the two low-order bits of the STA register is set.

One frame of data is 496 bytes if TR_EN is clear, or 512 bytes if TR_EN is set. The DAT register must be read 496 or 512 (respectively) times to retrieve one image frame.

Check the status register. If the two low-order bits are all set, then another A/D conversion is not necessary.

If the Busy flag is set, then an A/D conversion operation is already in progress; therefore, it is not necessary to start either a gain adjustment operation or an A/D conversion operation.

The GetImage function of the example source code has a variable that specifies when the gain should be re-adjusted.

This starts the automatic gain control operation.

The Busy flag will set during the automatic gain control operation. Wait until the Busy flag clears.

For efficiency, start another A/D conversion operation so that the sensor can be doing an A/D conversion while the host is processing the image data.

Data is packed two nibbles per byte. The upacking operation separates the two nibbles into two bytes.

Figure 14. Get Image Operation Flowchart