

Application Note

AN2382/D
Rev. 0, 11/2002

Integrating the
MCF5249 to a Biometric
Fingerprint Sensor



Christopher Baker
TECD Applications

Biometric technologies use a wide variety of physiological or behavioral characteristics, such as a person's fingerprint, hand, voice, iris, face, keystroke, or signature, to verify an individual's identity. Fingerprint comparison is the most cost effective and widely used method of biometric authentication. It is particularly suited to security applications because the chance of two people, even identical twins, having the same fingerprint is less than one in a billion. Fingerprint biometrics can be implemented in various security system designs ranging from personal computer password replacement, to basic, stand-alone verification systems for restricted access areas, to a multi-unit networked system with a central hub and many remote scanning units that could be used in settings such as an airport.

The purpose of this application note is to detail the hardware design and software development of a reference design which interfaces the Motorola ColdFire® MCF5249 microprocessor to the Fujitsu MBF200 fingerprint sensor. The reference design takes the form of a stand-alone restricted access system; however, this basic system can be applied as a complete stand-alone package or as one part of a larger solution.

This application note is intended to accompany the schematics, drivers, and example software available on the ColdFire web site as part of the complete reference design.

1 Design Overview

The object of this section is to outline the principles of the MCF5249 Biometrics reference design, and to give an overview of the MCF5249 microprocessor, the MBF200 fingerprint sensor, the Palm Technology PSG7955AW LCD display, and the Acter AG Biometric fingerprint algorithm. For more detailed information on the MCF5249, the MBF200, the PSG7955AW, and Acter AG software licensing please visit the following web pages:

<http://www.motorola.com/coldfire>

<http://www.fme.fujitsu.com/products/biometric>

<http://www.palmtech.com.tw>

<http://www.acter.net>

Figure 1 shows the basic block diagram of the reference design. The MCF5249 is interfaced to the MBF200 sensor using the external bus interface, and to the PSG7955AW LCD module using the queued serial peripheral interface (QSPI). Flash memory on the M5249C3 evaluation board is used as fingerprint template storage. The sensor provides a digital fingerprint image to the MCF5249, which runs the biometric algorithm provided by Acter AG. The LCD provides information such as "enrollment complete" or "verification failed" to the user.

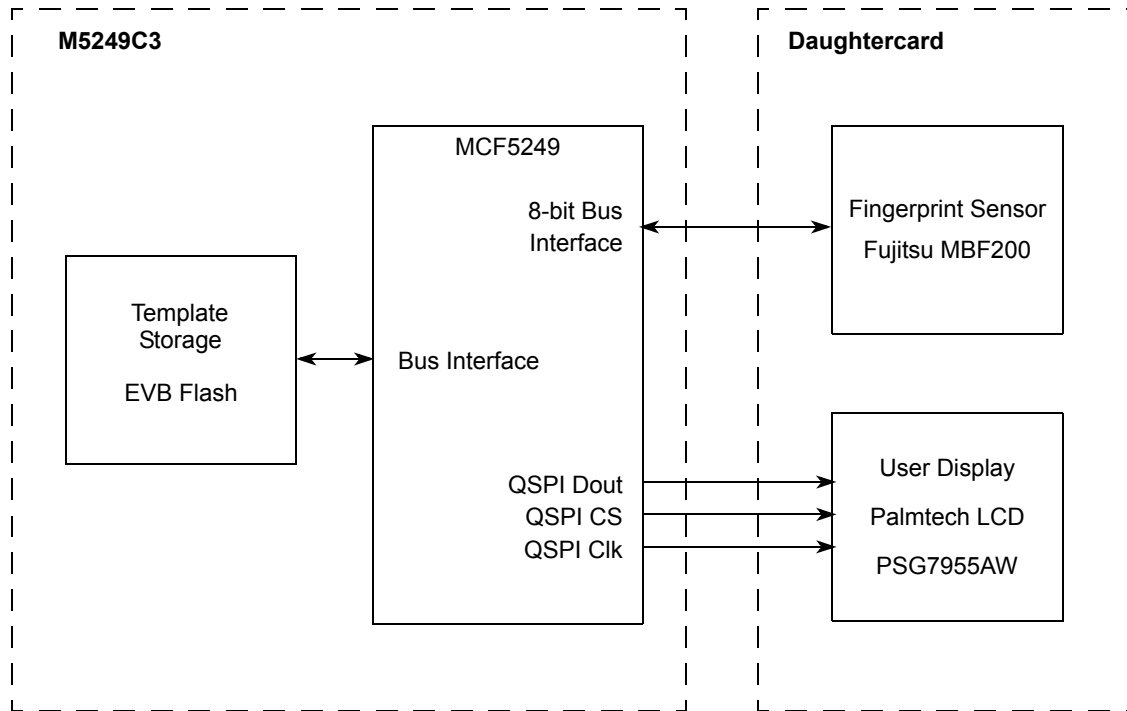


Figure 1. MCF5249 Biometrics Reference Design Block Diagram

1.1 MCF5249 Microprocessor

The MCF5249 is a 32-bit embedded microprocessor based on a V2 ColdFire core. This is the highest performing V2 product to date, with 125 Dhrystone MIPS at 140MHz. It has been designed to operate with low power consumption, using a 1.8V core and 3.3V I/O while consuming just 1.4mW per MHz with a wide variety of on-chip peripherals.

These on-chip peripherals include a DMA controller, integrated enhanced multiply accumulate unit (EMAC), 8 Kbyte direct-mapped instruction cache, 96 Kbyte SRAM, various audio interfaces, IDE, Smartmedia and Flash card interfaces, a QSPI module, an analog/digital converter, and two I²C modules. It also retains the system integration module, chip select module, general purpose timers, and the real-time BDM interface that are standard on all ColdFire devices.

The decision to create this reference design around the MCF5249 ColdFire microprocessor was based on the high performance of the core with the EMAC unit, low power consumption, the peripheral set, and overall system cost. With stand-alone biometric identification and verification systems there is a need for low cost and low power while still providing high performance in order to process the biometric matching algorithms.

The on-chip cache and SRAM allow the MCF5249 to run algorithms internally for the highest performance while only accessing external memory when extracting, comparing, and verifying the fingerprint templates. The various storage interfaces, such as IDE and Flash media, provide a versatility and flexibility in system design that makes the MCF5249 suitable for a wider variety of solutions.

1.2 Fingerprint Sensor

The Fujitsu MBF200 solid-state fingerprint sensor is a direct contact fingerprint acquisition device. It is manufactured using standard CMOS technology and has a 256x300 pixel, 50 μ m pitch sensor array that yields a 500 dpi image. Each pixel is a metal electrode that acts as one plate of a capacitor with the contacting finger acting as the second plate. A passivation layer on the device surface acts as the dielectric between these two plates. The ridges and valleys of the finger provide differing capacitances, and the corresponding voltage discharges are read to form the fingerprint image.

The MBF200 has an on-board analog to digital converter, an SPI interface, an 8-bit bus interface, and a USB interface, allowing it to minimize extra circuitry by providing the microprocessor with the digital image directly. It is a low power device running at either 3.3 or 5V and consuming less than 70mW at 5V, which makes it suitable for portable and remote applications.

Though there are several optical sensors, and other capacitive sensors, on the market, the MBF200 was chosen because it has a high throughput when using the 8-bit bus interface, making it ideal for interfacing to the MCF5249 (10 fps with SPI, 20 fps with USB and 30 fps with the 8-bit bus interface). Another attractive feature is the sensor's low power operation, which keeps the system power consumption to a minimum. The MBF200 also has an on-board analog to digital converter. This allows the sensor to process the analog voltage and provide the digital version directly to the microprocessor, therefore reducing the processing overhead and increasing the performance of the algorithm. As the 8-bit bus interface provides the highest image data throughput from the sensor, it has been used to connect to the MCF5249 in the reference design.

1.3 LCD Module

The Palm Technology PSG7955AW LCD module is a complete LCD and controller in one unit. It has a Supertwist Nematic (STN), positive, reflective type LCD with 132x40 pixels in its array. Built into the module is the Samsung S6B1713 LCD driver which provides all of the common and segment signals necessary to control the display. The module only brings out certain pins from the controller, providing the rest internally. The available pins are the voltage converter pins, the supply pins, chip select, reset, and the SPI interface pins. The data out pin for the SPI interface is not available internally; therefore, the LCD module functions in a write-only mode.

The controller has an internal RAM block that directly maps to the LCD display itself. Once the module has been set up correctly with the display switch on, any bit set in the RAM will be mirrored by its corresponding LCD pixel. The controller operates by receiving 8-bit commands and acting accordingly. Examples of these commands are Entire Display On, Reverse Display, Write Display Data, and Reset.

1.4 Biometric Fingerprint Algorithm

Acter AG provides the biometric fingerprint algorithm used in the example image file. This algorithm is minutia-based and runs through three different operating modes: Enrollment, Matching, and Verification. Fingerprint minutia are defined either as the ends of a ridge or the join of two ridges. Figure 2 shows a fingerprint image with the ridges and minutia highlighted.

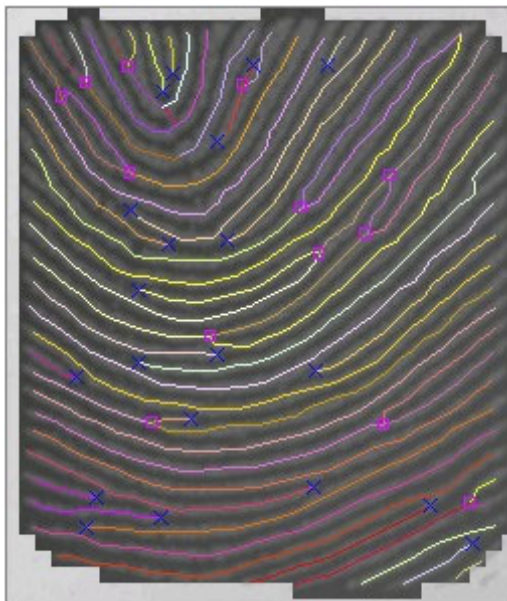


Figure 2. Fingerprint Image Showing Ridges and Minutia

1.4.1 Fingerprint Enrollment

At the beginning of this process, the algorithm scans the fingerprint image for minutia. (Since the algorithm will reject the image if there are not enough usable minutia in the image, the quality of the sensor image is important.) The user will be asked to place his or her finger on the sensor several times while the algorithm takes several templates and uses these to create a master template for that finger. Each minutia found is given a weighting based on the number of times they matched for each of the successive templates. Accidental or single-match minutia are dropped creating a final template of very high quality. This procedure makes future comparison easier and more accurate, while increasing the inherent level of security by reducing the chance of an incorrect match.

Note that the system does not store the actual fingerprint image, but only the template created from the minutia found in the image. This increases the security of the system and prevents someone from accessing the stored data and stealing a user's identity. Once the template has been completed and stored, the user has been enrolled in the system and can use the matching and verification modes.

1.4.2 Fingerprint Matching

In order to match a fingerprint, the user is asked to place his finger on the sensor again. The fingerprint image is extracted from the sensor and the minutia in this image are compared to those stored in the template. Figure 3 shows a fingerprint image with the minutia selected for matching. This vectored pattern of minutia is compared to the patterns within the template and corresponding minutia are noted. With the algorithm provided by Acter AG, the orientation requirements of the finger or vector pattern is eliminated as the algorithm can accommodate rotational and translational variances.

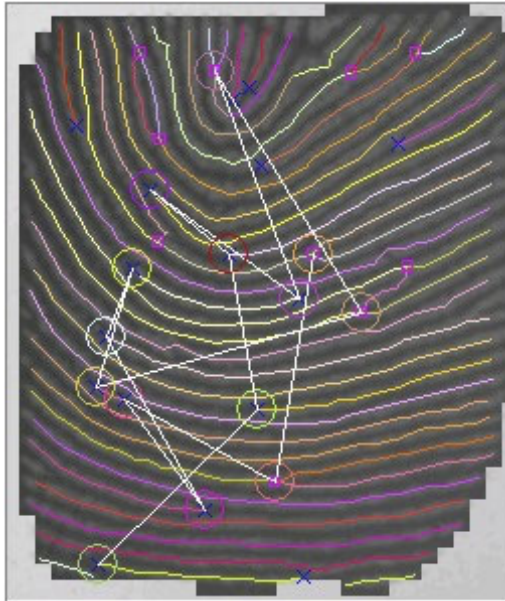


Figure 3. Fingerprint Image Showing the Matching of Minutia

1.4.3 Fingerprint Verification

Within the algorithm, a set threshold determines when enough minutia from the fingerprint image match those in the fingerprint template. If this threshold is passed, then the fingerprint image is determined to be of the same finger that created the template. Modification of this threshold can result in a wide variety of accuracy. The threshold value, along with the algorithm's verification speed, is used to determine the algorithm's overall performance. If the threshold is lowered, then the chance of incorrectly rejecting the correct finger, known as the False Rejection Rate (FRR), is also lowered. At the same time, lowering the threshold also increases the chance of incorrectly accepting the wrong finger, known as the False Acceptance Rate (FAR). Where the FAR and FRR are equal is termed the Equal Error Rate (ERR), and is shown in Figure 4.

Moving the threshold level and the ERR to the lowest and highest extremes will lower the accuracy of the algorithm and compromise the system's efficacy: the lowest extreme will allow anyone to be verified regardless of enrollment, and the highest extreme will completely block anyone from ever being verified. The goal is to set the threshold at a level that maintains security and also allows enrolled users to be easily recognized.

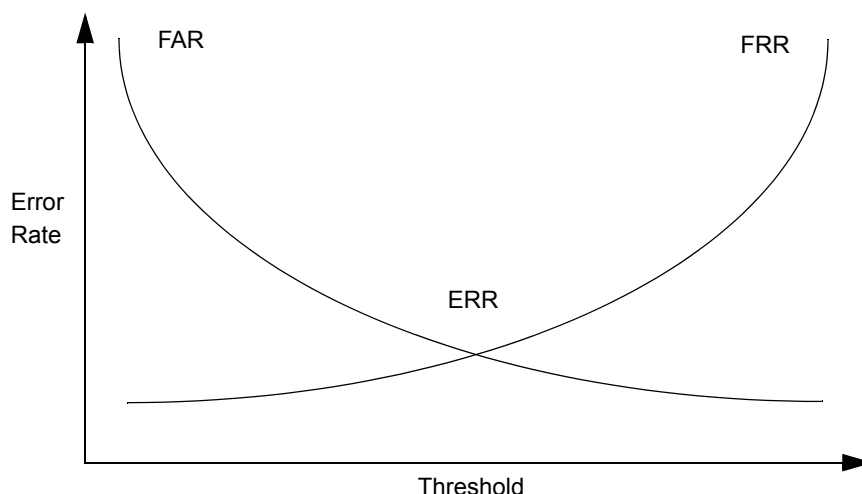


Figure 4. FAR, FRR and ERR Comparison to the Threshold Level

1.4.4 Fingerprint Template

The fingerprint template for a minutia-based algorithm is created from the minutia extracted from the image. This has several additional bonuses in that the template size is small, and the security of the system is increased. For other algorithms, the template is the fingerprint image, which, for the MBF200, is 75 Kbytes. In contrast, the minutia-based algorithm template is between 250 Bytes and 2 Kbytes, thus reducing the amount of non-volatile memory required by the system and increasing the speed of the algorithm. In order to increase the acceptability of the system, users need to be certain that their identification cannot be stolen. The minutia-based template can never be used to re-create the fingerprint itself, thereby removing the possibility of theft.

2 Hardware Design

The MCF5249 Biometrics reference design is based around the M5249C3 evaluation board and utilizes a daughtercard to connect the sensor and LCD to the microprocessor via the on-board expansion connectors. The board also provides the 10/100 ethernet interface, RS232 interface, BDM interface, 8 Mbytes SDRAM and 2 Mbytes of Flash ROM for system development and test. For additional information on the M5249C3 board, including full schematics, refer to the users manual available from the MCF5249 Biometrics web page.

The rest of this section explains key features of the reference design: the QSPI interfacing, the external bus interfacing and timing, LCD power grid, and interrupt options. Schematics of the daughtercard are also available from the MCF5249 Biometrics web site.

2.1 LCD Interface

An on-board SPI interface was one reason for selecting the PSG7955AW LCD module. This feature allows the module to directly connect to the QSPI interface on the MCF5249 which results in efficient data transfer and easy configuration.

2.1.1 SPI Module Interface

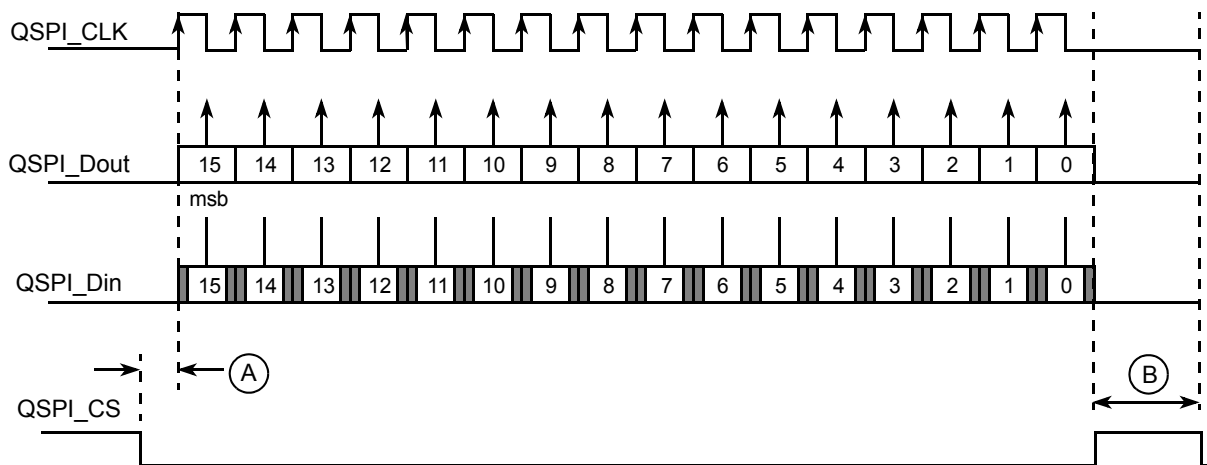
The QSPI module on the MCF5249 provides a serial peripheral interface with queued transfer capability which allows up to 16 data transfers with no CPU intervention. The QSPI interface will support data transfers, msb first, of between 8 and 16 bits. It will support baud rates of 235.3 Kbps to 15 Mbps and can be interfaced to a maximum of 15 devices using the 4 peripheral chip-select lines.

The module has a total of seven signals: serial data output (QSPI_Dout), serial data input (QSPI_Din), QSPI clock output (QSPI_CLK) which is used by the external devices to synchronize the message reception, and the four dedicated peripheral chip selects (QSPI_CS[3:0]). Interface to the LCD module in the reference design uses the following four signals: QSPI_Dout, QSPI_CLK, QSPI_CS0 and QSPI_CS1.

Clock phase and polarity, chip select active logic level, and delays before and after transfer, highlighted in Figure 5, are all programmable via the QSPI registers. This flexibility in clocking and data transfer eliminates the need for additional glue logic to meet the PSG7955AW timing requirements.

The SPI interface on the LCD module is write only. That is, the data out pin is not available, making it a four signal interface: serial data input (SID), serial clock input (SCLK), register select input (RS), and chip select input (CS). The RS signal is used by the LCD module to determine whether the data coming in on the bus is display data, indicated by a high level, or control data, indicated by a low level. This is generated using the MCF5249 QSPI_CS1 signal.

The QSPI is programmed to use the default delay values and the BAUD bits in the QMR register are set to 2, resulting in an SPI clock of 17.5MHz



A = QSPI clock delay. Programmed in the QSPI delay register, QDLYR.

B = Delay after transfer. Programmed in the QSPI delay register QDLYR.

Clock polarity is set to 0, making the inactive state of QSPI_CLK logic level 0.

Clock phase is set to 1, that is, data changed on the rising edge and captured on the falling edge.

Figure 5. QSPI Clocking and Data Transfer Parameters

2.2 Fingerprint Sensor interface

Several interfaces are built into the Fujitsu MBF200, the USB interface, the SPI interface, and the parallel bus interface. The best performance of the sensor is obtained by interfacing through the parallel bus where the microprocessor can receive 30 frames per second (compared to 20 for the USB and 10 for the SPI).

2.2.1 Parallel Bus Interface

The external bus interface on the MCF5249 has a 23-bit address bus with a programmable port size of 8 or 16 bits. It can generate byte, word, and longword transfers, supports burst and burst-inhibited transfers, and can generate internal termination signal. There are four independent chip selects capable of accessing external peripherals. For the reference design, CS3 has been chosen to ensure that the sensor interface will not interfere with other peripherals already connected on the M5249C3 board (such as the Flash on CS0 and the ethernet controller on CS1).

Due to the multiplexed nature of many of the pins on the MCF5249, pin operations are selected as shown in Table 1 to generate the required $\overline{\text{CS}}$, $\overline{\text{WR}}$, and $\overline{\text{RD}}$ signals. Table 1 also shows the connection to the sensor pins.

Table 1. MBF200 Sensor Interface Pin Descriptions

MCF5249 pin	Selection	Description	MBF200 pin
$\overline{\text{BUFENB1}}/\text{GPIO57}$	$\overline{\text{BUFENB1}}$	Used as the chip select	$\overline{\text{CS0}}$
$\overline{\text{SWE}}/\text{GPIO12}$	$\overline{\text{SWE}}$	Used as the write enable	$\overline{\text{WR}}$
$\overline{\text{CS3}}/\overline{\text{SRE}}/\text{GPIO1}$	$\overline{\text{SRE}}$	Used as the read enable	$\overline{\text{RD}}$

To generate the write enable and read enable signals required by the sensor, the $\overline{\text{SWE}}$ and $\overline{\text{SRE}}$ signals are used. These signals are selected by setting the `bufen1cs3en` bit in the `IDECONFIG1` register and by setting up the CS3 registers to the appropriate address space. This process is discussed further in Section 3.2, “MCF5249 Initialization.”

The 8-bit MBF200 parallel bus interface uses read and write enables, two chip select lines, and a single address line to control the read and write data transfers. The MBF200 uses indirect addressing to access its internal registers, only allowing direct access to the index register and the data register. The truth table shown in Table 2 illustrates data transfer modes for various signal conditions. As this table shows, the address line A0 is used to determine whether the data on the bus is intended for the index register or the data register, thereby facilitating the indirect addressing.

To implement this truth table using the $\overline{\text{SRE}}$, $\overline{\text{SWE}}$ and $\overline{\text{BUFENB1}}$ signals, the CS1 pin on the sensor has been tied to 3.3 V (logic level 1), and the MCF5249 address line A1 has been connected to the sensor A0 line. This means that, to access the 2 available sensor registers, the MCF5249 has to write to or read from address `0xAAAAbbbC`, where AAAA represents the value placed into the CS3 address mask bits [15:0] in register `CSMR3`, b = don't care, and C selects the register as indicated below:

Index register: C = xx0x in binary (x = don't care)

Data register: C = xx1x in binary (x = don't care)

CS3 has been programmed with a mask of `0x5000` and the addresses selected to access the Index and Data registers are `0x50000000` and `0x50000002` respectively.

Table 2. Truth Table for MBF200 Bus Interface

$\overline{\text{CS0}}$	CS1	A0	$\overline{\text{RD}}$	$\overline{\text{WR}}$	Mode	Data Lines
H	X	X	X	X	De-selected	High Impedance
X	L	X	X	X	De-selected	High Impedance
L	H	X	H	H	Standby	High Impedance
L	H	L	L	H	Read Index Register	Output

Table 2. Truth Table for MBF200 Bus Interface (continued)

$\overline{\text{CS0}}$	CS1	A0	$\overline{\text{RD}}$	$\overline{\text{WR}}$	Mode	Data Lines
L	H	L	H	L	Write Index Register	Input
L	H	H	L	H	Read Data Register	Output
L	H	H	H	L	write Index Register	Input

2.2.2 Parallel Bus Timing

The MCF5249 external bus can insert from 2 to 132 wait states to allow connection to a wide variety of peripherals. These are programmed via the CSCR3 register bits WS[1:0] for chip select 3. With the MBF200 running off of a 12-MHz crystal and the MCF5249 running at 140 MHz, two wait states are required to meet the timing requirements of the sensor. The specification on the MBF200 is flexible enough to accept the control signals from the MCF5249 with two wait states as the only required adjustment.

2.3 Miscellaneous

The MBF200 has a built-in automatic finger detect function that can generate an interrupt when it detects that a finger has been placed on or removed from the sensor. This is a unique feature that allows the system to wake up from a low power mode upon detection, start the enrollment or verification sequence, and return to energy saving mode upon completion.

The resistors and capacitors around the MBF200 sensor are chosen based on the recommended values in the sensor specification. Of special note is the 56 K Ω resistor attached to the FSET pin. Recommended for 12 MHz operation, the resistor sets the internal multi-vibrator and the automatic finger detect frequency. Inputs tied high are connected through a resistor to improve ESD and noise immunity for the sensor's capacitive array.

The MODE pins on the sensor are connected to ground to set the sensor in microprocessor bus mode.

The INTR and EXTINT pins are connected to GPIO pins for future use in the reference design.

The variable resistor, R1 is used to control the operating voltage of the LCD by varying the voltage present at pins V0 and VR. In the application this can be used to adjust the contrast of the display.

The LCD module requires a power circuit connected to its voltage converter to boost the voltages for the display. The circuit used is a x3 boosting circuit as shown in Figure 6.

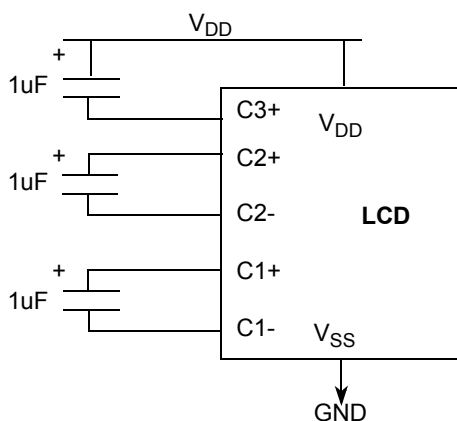


Figure 6. Voltage Converter Circuit

3 Software Development

This section discusses the software drivers and code used by a basic system example. These drivers and code display information for the user, read the fingerprint image from the sensor, and run a biometric algorithm function. It begins by providing an overview of the software flow from a system perspective and then goes on to outline how the QSPI interface is used to program the LCD module registers. This is followed by a description of the character set used to generate bitmap characters on the LCD display. The discussion then covers the interface to the sensor and how the sensor registers are read and written using the 8-bit parallel bus interface. Finally, it discusses the sensor timing requirements, ambient humidity considerations and sensor array capacitive discharge values, and includes a description of the example software main() function.

NOTE

This section does not cover the development of a biometric fingerprint algorithm. In the source example, the algorithm function is empty, while the limited binary executable uses the algorithm developed and owned by Acter AG.

The source code for the drivers and example system has been developed using Wind River's Diab compiler and VisionClick debugger; it can be downloaded from the MCF5249 Biometrics web page for evaluation and testing or for use as a template for further development. A limited executable image, including Acter AG's biometric fingerprint algorithm, is also available on the website and can be used as a demonstration of the MCF5249 and the algorithm's capabilities.

3.1 System Overview

From a software perspective, the first operation is to initialize the microprocessor, sensor, and LCD module. This involves setting up the MCF5249 operating parameters and configuring the external bus and QSPI interface modules. Once initialization of the microprocessor is complete, the sensor and LCD are configured for the required operation before the first message is given to the user via the LCD display. This could be a welcome message followed by a request for the user to place his finger on the sensor to begin enrollment.

The system then goes into a waiting loop while it waits for the detection of a finger on the sensor. When a finger is recognized, the LCD displays an "enrollment in process" message while the sensor captures the fingerprint image and the microprocessor extracts it. The biometric algorithm then enrolls the finger by creating a fingerprint template from the extracted minutia.

Next, the LCD displays a message informing the user that the enrollment has been completed, or if it failed, that enrollment will have to take place again. If the finger is enrolled, the LCD displays a message indicating that fingerprint verification will take place, and asking the user to place his finger back on the sensor. The algorithm extracts minutia from the second fingerprint image and compares this to the template stored during the enrollment phase.

Finally, the minutia are matched and the LCD displays a message indicating that access is granted (for a match), or access is denied (for a failed match). If the match failed, the user will be asked to replace his finger on the sensor, or to enroll himself into the system before it goes back into a waiting loop.

See Figure 7 for a graphical representation of the system flow.

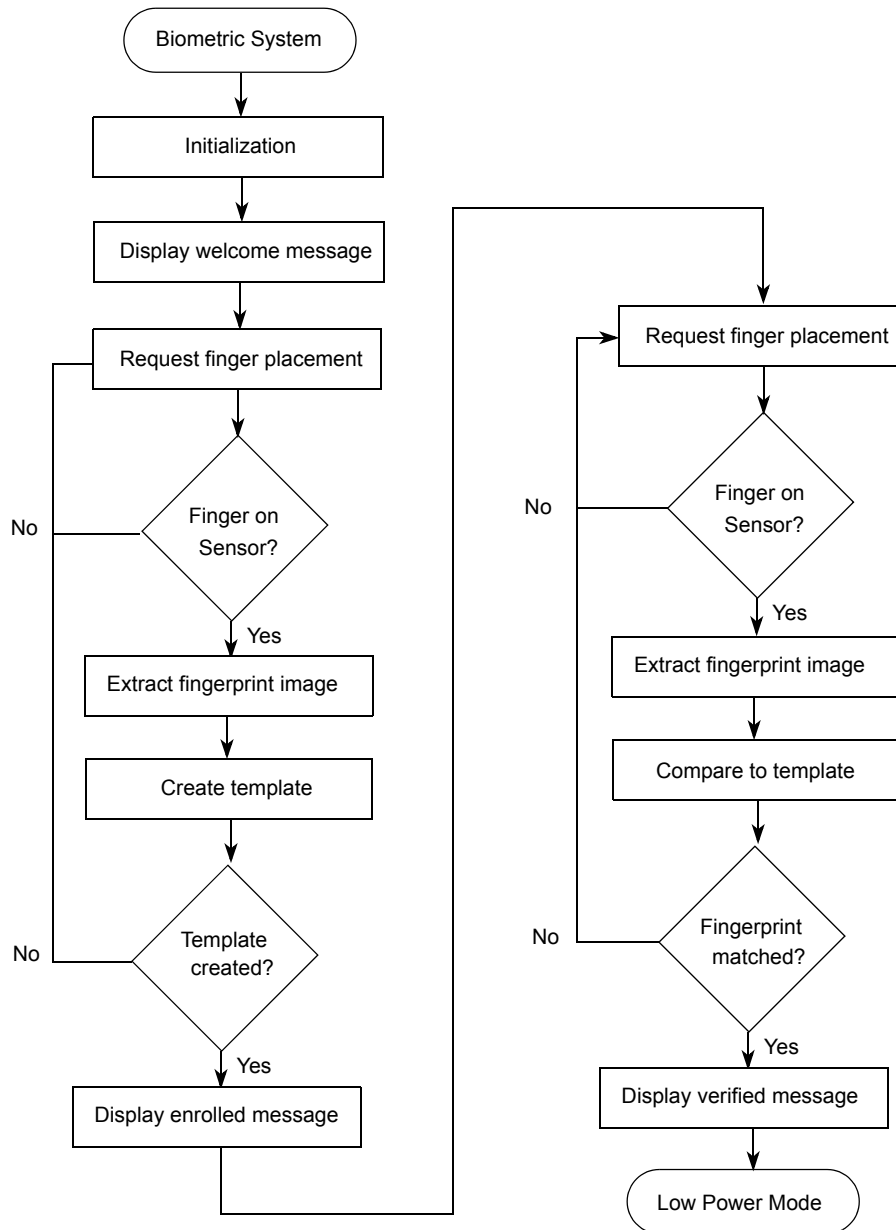


Figure 7. Basic System Flowchart

3.2 MCF5249 Initialization

The MCF5249 is set up to run at 140 MHz, from the 11.2896 MHz crystal. This provides the highest performance for the algorithm execution, by programming the PLLCR register with the value 0x01af6837. This sets the CPU clock divider to 1, the PLL compare frequency divider to 246, the PLL divider to 3, and the VCO output divider to 1.

Note that after the PLLCR register is written, a loop ensures that the PLL is locked before continuing. The PLL should be programmed as soon as possible in order to get the CPU running at the desired frequency as soon as possible.

The significant MCF5249 peripheral modules used in this process are the external bus interface and the QSPI interface; this allows the CPU to communicate with the sensor and the LCD module. The following section of code is used to initialize the external bus interface control signals and the pins used for the interface:

```

/*--- Chip select registers ---*/
MCF5249_CS_CSAR3 = 0x5000; //Base address of memory mapped to CS3
MCF5249_CS_CSMR3 = 0x00000001; //Mask for CS3 address decode plus valid bit
MCF5249_CS_CSCR3 = 0x00000941; //2 wait states, 8-bit port and valid bit
MCF5249_IDE_CONFIG1 = 0x00040000; //BUFENB1 active on CS3 (SRE/SWE cycles)
MCF5249_IDE_CONFIG2 = 0x00010000; //TA generation on CS3

/*--- Sensor Bus Signals -WR -RD -CS0 setup ---*/
MCF5249_SIM_GPIO_FUNC &= ~0x00000800; //Sets the pin function to -SRE
MCF5249_SIM_GPIO_FUNC &= ~0x00001000; //Sets the pin function to -SWE
MCF5249_SIM_GPIO1_FUNC &= ~0x02000000; //Sets the pin function to -BUFENB1

```

The CSAR3 and CSMR3 chip select registers are programmed to mask off the address space with a base address of 0x50000000. The CSCR3 register sets the port size to 8-bit, inserts two wait states into the bus accesses on the CS3, and sets the chip select register valid bit. The SRE, SWE and BUFENB1 pins are multiplexed with other functions; therefore, the GPIO registers have to be programmed to select the correct function for the pins.

This next section of code initializes the QSPI registers to enable the correct function of the QSPI module to communicate with the LCD module. It also sets up the general purpose output pin 24 for use as the LCD reset signal.

```

/*--- QSPI registers ---*/
MCF5249_PLL_PLPCR |= 0x00000800; //Sets the pin function to QSPI_CLK,
MCF5249_SIM_GPIO_FUNC &= ~0x25000000; //QSPI_Dout, QSPI_CS0 and QSPI_CS1
MCF5249_QSPI_QMR = 0xC302; // MSTR, DOHIE, DIVIDE BY 2

/*--- Setup the MCF5249 port pins for the LCD interface ---*/
MCF5249_SIM_GPIO1_FUNC |= 0x00000020; //Sets the pin function to output
MCF5249_SIM_GPIO1_OUT &= ~0x00000020; //Holds the LCD module in reset
MCF5249_SIM_GPIO1_ENABLE |= 0x00000020; //Enables the output
MCF5249_QSPI_QMR = 0xC302; // MSTR, DOHIE, DIVIDE BY 2

```

Note that the QSPI has multiplexed pins; in order to select the QSPI function the QSPISEL bit in the PLLCR register has to be set. The QSPI_Dout, QSPI_CS0 and QSPI_CS1 pins are also multiplexed with other functions; therefore, the GPIO register has to be written to select the correct function for the pin. Finally, the QMR register is written to set the QSPI to master mode, the data lines to high impedance when not driven, the QSPI_CLK signal to be active low, and data to be captured on the falling edge of the clock.

The LCD module requires that a specific delay be inserted between certain commands issued as part of its power-up sequence; therefore, Timer0 has been used to count for a 5 μ s count, utilizing an interrupt routine to increment a tick every 5 μ s. The interrupt function increments the variable delayTick and then resets the timer to start counting again for another 5 μ s. The delay() function shown below is passed one parameter which indicates how many times the delayTick should be incremented by the timer1 interrupt routine.

```

/*--- 5us delay routine using Timer0 ---*/
void delay (uint32 delay)
{
    uint32 tester = 0;

    MCF5249_SIM_ICR1 = 0x84; // set Timer0 to Interrupt level 1, priority 0
    MCF5249_SIM_IMR = 0xFFFFDFFF; // enable timer0 interrupts
/*--- Setup timer for a 5us count ---*/
// master clock /16, prescaler 1, interrupt on TRR, enable timer
    MCF5249_TIMER0_TMR = 0x001D;
    MCF5249_TIMER0_TRR = 0x0016; // timer reference value (22 = 0x16)
    delayTick = 0;
    while(delayTick<delay)
        tester = (delay - delayTick);
    MCF5249_TIMER0_TMR = 0; //reset timer
}

```

Finally, the following code is used to set up a general purpose input pin that is connected to the switch used by the user to select enrollment or verification mode.

```

/*--- Enrollment/Verification switch setup ---*/
MCF5249_SIM_GPIO1_FUNC |= 0x00100000; //Sets the pin function to GPIO52
MCF5249_SIM_GPIO1_ENABLE &= ~0x00100000; //Sets the GPIO to input

```

3.3 LCD Interface

The QSPI interface is used to communicate with the LCD module utilizing two of the QSPI_CS signals, the QSPI_Dout, and QSPI_CLK. The QSPI module is initialized as described previously; however, it is important to note that the SPI interface on the LCD module is write only and that there are no QSPI read functions in the example software. The WriteLCDByte() function is used as the basic driver for the LCD interface.

3.3.1 WriteLCDByte()

Two parameters are passed into the WriteLCDByte() function. One is the data to be transferred to the LCD module and the other is the desired RS signal value, which defines whether the data is intended for display data or is a command. Note that there is a loop that delays transmission of a second message until the first message transmission is complete.

The QSPI command is created using the RS parameter to set the QSPI_CS1 pin high or low according to requirements, and to set the QSPI_CS0 high during transfer. Note that only 1 byte is transferred to the LCD per QSPI message.

```

/*--- Write a byte into the LCD ---*/
void WriteLCDByte(uint8 data, uint8 rs)
{
    uint16 tmp;
    uint16 qspiCommand = 0x0100; // QSPICS0 active high

    if(rs)
        qspiCommand |= 0x0200; // set QSPICS1 to match rs
    while(MCF5249_QSPI_QDLYR & 0x8000)
        ; // wait for any previous transmit to complete
    MCF5249_QSPI_QAR = 0x0020; // command buffer
    MCF5249_QSPI_QDR = qspiCommand;
    MCF5249_QSPI_QAR = 0x0000; // transmit buffer
    MCF5249_QSPI_QDR = (uint16)data;
    MCF5249_QSPI_QDLYR |= 0x8000; // initiate transfer
}

```

3.3.2 LCD Initialization

In order to be fully functional in the system, the LCD module has to be initialized and powered up in the correct sequence. The power-up sequence involves issuing several commands to the LCD controller, beginning with resetting the LCD module. Next, the column direction should be set to normal, the row direction to normal, the LCD duty ratio to 1/49, and LCD bias ratio to 1/8. These steps are performed before powering up the voltage converter, regulator, and follower circuits in that order. The code below shows the commands sent to the LCD module using the WriteLCDByte() function. Note that there is a minimum of 1ms delay inserted between each voltage circuit being switched on and the next one.

```

/*--- SET ADC, SHL AND LCD BIAS ---*/

WriteLCDByte(LCD_ADC, LCD_CTRL);
WriteLCDByte(LCD_SHL_NORMAL, LCD_CTRL);
WriteLCDByte(LCD_BIAS_SELECT, LCD_CTRL); // 1/49 DUTY ; 1/8 BIAS

/*--- CORRECT POWER SEQUENCE, SWITCH ON VOLTAGE CONVERTER, THEN REGULATOR,
THEN FOLLOWER, WITH 1msec MINIMUM DELAY BETWEEN EACH STAGE ---*/

WriteLCDByte(LCD_PWR_CTRL_VC, LCD_CTRL);
delay(200);
WriteLCDByte(LCD_PWR_CTRL_VR, LCD_CTRL);
delay(200);
WriteLCDByte(LCD_PWR_CTRL_VF, LCD_CTRL);
delay(200);

```

Finally the voltage regulator resistor is selected and the reference voltage select commands are used to complete the LCD power-up sequence.

wide. In the function call, the numeric values after the string of text define which row and column of the LCD array to start the text.

The VarPrint() function works in a similar way with an extra option to display the variable in decimal, 8-bit hexadecimal, or 16-bit hexadecimal.

3.5 Sensor Interface

The MBF200 fingerprint sensor uses indirect addressing to access its registers through the 8-bit parallel bus. There are two registers directly available via the bus: the index register and the data register. The address offset of the desired internal register is written to the Index register, then the data is either read from the Data register or written to it as required. See section 3.6 MBF200 Sensor Register Map for information on the sensor registers.

3.5.1 FPWriteReg()

The FPWriteReg() function has two parameters passed to it: the offset of the register required to be written and the data to be written to that register. Both of these 8-bit parameters are written into the sensor using the FPWriteIndexReg and FPWriteDataReg macros. The sensor macro definitions and FPWriteReg() function are shown below. For more information on the MBF200 register offsets, see Section 3.6, “Sensor Register Map.”

```
//Write to the INDEX register
#define FPWriteIndexReg(REG) (cpu_iowr_8(FP_INDEX_REG,REG))
//Write to the DATA register
#define FPWriteDataReg(DATA) (cpu_iowr_8(FP_DATA_REG, DATA))
void FPWriteReg(uint8 REG, uint8 DATA)
{
    FPWriteIndexReg(REG);
    FPWriteDataReg(DATA);
}
```

3.5.2 FPReadReg()

The FPReadReg() function has one parameter passed to it which is the offset of the register to be read from, and returns the data read from the register. This register offset is written to the sensor using the FPWriteIndexReg macro before the FPReadDataReg macro is used to read the data back from the register. These in turn call standard ColdFire write and read macros as defined in the mcf5249.h header file. The sensor macro definitions and FPReadReg() function are shown below. For more information on the MBF200 register offsets see Section 3.6, “Sensor Register Map.”

```
//Write to the INDEX register
#define FPWriteIndexReg(REG) (cpu_iowr_8(FP_INDEX_REG,REG))
//Read from the DATA register
#define FPReadDataReg() (cpu_iord_8(FP_DATA_REG))

uint8 FPReadReg(uint8 REG)
{
    FPWriteIndexReg(REG);
    return(FPReadDataReg());
}
```

3.5.3 Sensor Initialization

The MBF200 fingerprint sensor has to be initialized correctly before it is fully functional. This includes enabling the sensor, setting the sensor array discharge time and current, setting the amplifier gain, setting threshold values for the auto detect interrupt, and enabling the interrupt. The values used for the discharge time, discharge current, and amplifier gain were provided by Fujitsu. The code shown below is used to setup the MBF200 sensor.

```
FPWriteReg(FP_CTRLA, 0x00);
FPWriteReg(FP_THR, FP_THV_STARTUP | FP_THC_STARTUP); //Setup threshold values
//Setup sensor control
FPWriteReg(FP_CTRLB, FP_ENABLE | FP_XTALSEL | FP_AUTOINCEN | AFDEN);
//Setup discharge values
FPAdjustParams(FP_DT_STARTUP, FP_DC_STARTUP, FP_GAIN_STARTUP);
FPWriteReg(FP_ISR, 0x03); //Clears any pending interrupts
//Enable finger detect interrupt
FPWriteReg(FP_ICR, FP_IP_FINGER | FP_IT_FINGER | FP_IE_FINGER);
```

3.5.4 Fingerprint Sensor Row Capture Timing

The sensor array functions by capturing capacitive discharge values from the sensor surface and the finger one row at a time. After the discharge values have been captured, the A/D converter is used to read off each pixel in the row one pixel at a time. The delay between starting and completing the row capture is calculated using the following formula:

$$\text{Row Capture Time} = (28 + \text{DT}[6:0]) * \text{sensor clock period}$$

where DT[6:0] are the discharge time bits in the DTR register.

For a sensor clock of 12 MHz and a DT value of 55, the resulting capture time is 6.9µs.

3.5.5 Fingerprint Sensor A to D Conversion Timing

The sensor has an auto-increment function that allows it to automatically convert the next array pixel after the data has been read from the A/D converter via the CTRLA register. This function has been activated in the initialization code above, and allows the sensor to convert the array data and produce the fingerprint image faster. As a result, when performing consecutive reads, the MCF5249 has to wait until the A/D

converter has completed its function before attempting to read the next data byte. The delay is calculated using the following formula:

$$\text{A/D Conversion Time} = 6 * \text{sensor clock period}$$

For a sensor clock of 12 MHz this equates to a conversion time of 500ns.

The next 3 sub-sections contain the `FPMCUGetRow()`, `FPMCUGetWhole()` and `FPMCUGetSub()` flowcharts and code. Each of these functions are image retrieval functions used to pull the digital image of a portion or the whole sensor array.

Note that the flowcharts and code for `FPMCUGetRow()`, `FPMCUGetWhole()` and `FPMCUGetSub()` include delays for the row capture time as well as the A/D conversion time.

The **`FPMCUGetRow()`** function is used to capture a single row of array pixels. It is passed three parameters: the structure that will hold the data read from the sensor, the upper row address byte, and the lower row address byte. The address is placed in the RAH and RAL registers.

The **`FPMCUGetWhole()`** function retrieves the data for the complete sensor array. No parameters are passed as the sensor will start at the first pixel in the first row and continue sampling and conversion until it has reached the last pixel in the last row.

The **`FPMCUGetSub()`** function retrieves the data for a section of the sensor array. This section is defined by the parameters passed into the function: the start row address, the start column address, the end row address, and the end row column. The sensor samples and converts every pixel from the start column on the start row until the end column is reached, then repeats the process for all the rows until the end row is reached. The addresses are placed in the RAH, RAL, CAL, REH, REL, and CEL registers.

3.5.6 FPMCUGetRow()

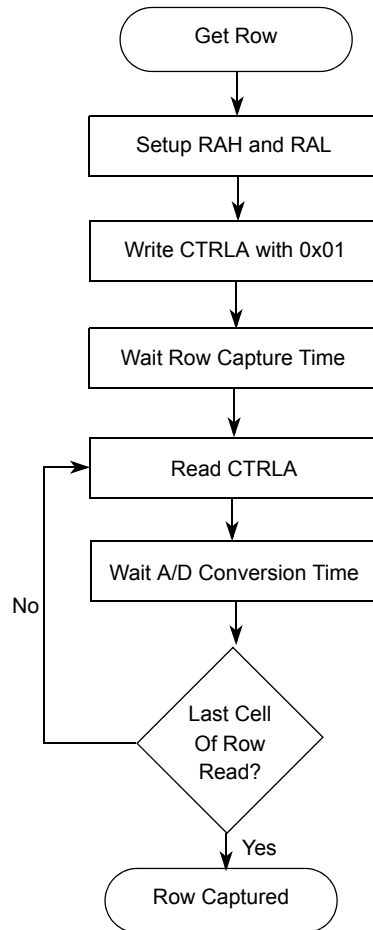


Figure 9. Row Capture Flowchart

```

/*--- MCU Get Row ---*/
void FPMCUGetRow(FP_ROW data, uint8 row_MSB, uint8 row_LSB)
{
    uint16 i;

    FPWriteReg(FP_RAH, row_MSB); //set MSB of row address
    FPWriteReg(FP_RAL, row_LSB); //set LSB of row address
    FPWriteReg(FP_CTRLA, FP_GET_ROW); //start row A/D sequence
    /*--- Wait Row Capture Time ---*/
    delay(2); //delay for 2 * 5us = 10us
    for (i=0; i<FP_MAX_COLUMN; i++) //loop for 256 pixels in row
    {
        data[i] = FPReadReg(FP_CTRLA);
        /*--- Wait A/D Conversion Time ---*/
        delay(2);
    }
}

```

3.5.7 FPMCUGetWhole()

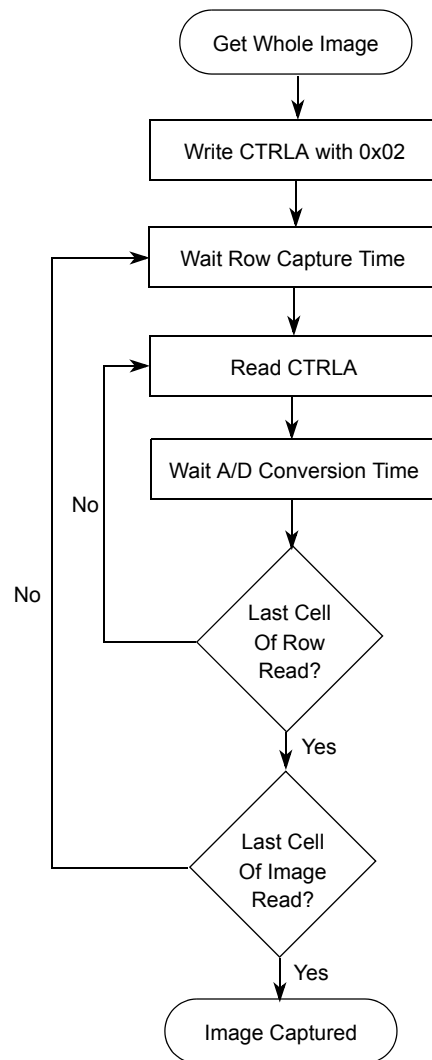


Figure 10. Whole Image Capture Flowchart

```

/*--- MCU Get Whole Image ---*/
void FPMCUGetWhole(FP_IMAGE data)
{
    uint16 i, j;
    uint32 reg1, reg2;

    FPWriteReg(FP_CTRLA, FP_GET_ROW); //start image A/D sequence
    for (j=0; j<FP_MAX_ROW; j++) //loop for 300 rows in whole image
    {
        /*--- Wait Row Capture Time ---*/
        delay(2); //delay for 2 * 5us = 10us
        for (i=0; i<FP_MAX_COLUMN; i++) //loop for 256 pixels in row
        {

```

```

data[j] [i] = FPReadReg(FP_CTRLA);
/*--- Wait A/D Conversion Time ---*/
delay(1);
    }
}
}
    
```

3.5.8 FPMCUGetSub()

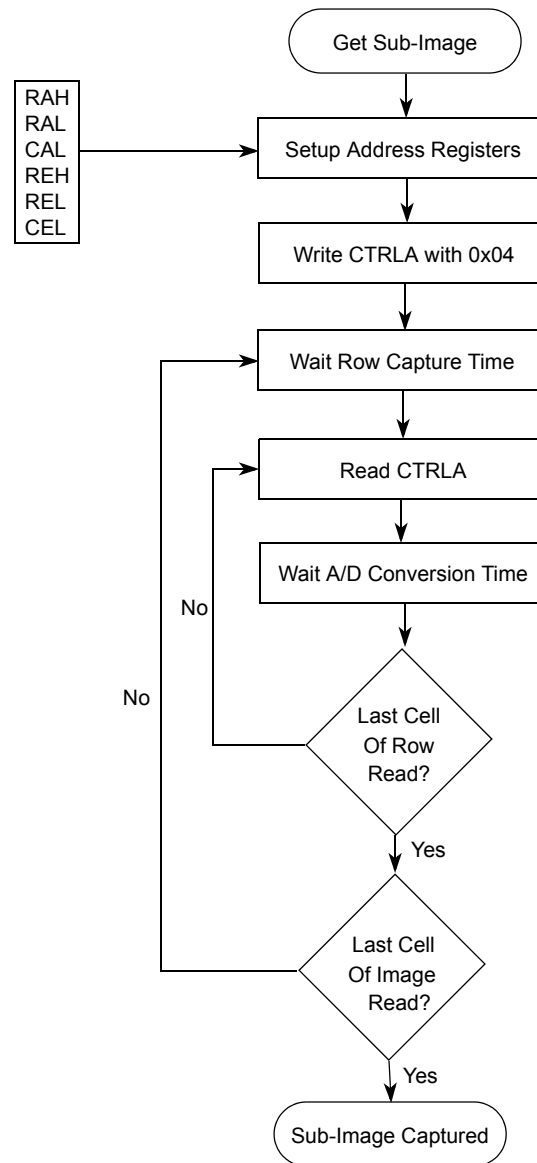


Figure 11. Sub-Image Capture Flowchart

```

/*--- MCU Get Sub-Image ---*/
void FPMCUGetSub(FP_IMAGE data, uint8 row_MSB, uint8 row_LSB, uint8 column,
    uint8 end_row_MSB, uint8 end_row_LSB, uint8 end_column)
{
    uint16 i, j, row_end;

    FPWriteReg(FP_RAH, row_MSB); //set MSB of row address
    FPWriteReg(FP_RAL, row_LSB); //set LSB of row address
    FPWriteReg(FP_CAL, column); //set column address
    FPWriteReg(FP_REH, end_row_MSB); //set MSB of end row address
    FPWriteReg(FP_REL, end_row_LSB); //set LSB of end row address
    FPWriteReg(FP_CEL, end_column); //set end column address
    FPWriteReg(FP_CTRLA, FP_GET_ROW); //start sub-image A/D sequence

    row_end = end_row_MSB;
    row_end = (row_end << 8) | (end_row_LSB);
    j = row_MSB;
    j = (j << 8) | (row_LSB - 1);
    for (j; j<row_end; j++) //loop for rows in sub-image
    {
        /*--- Wait Row Capture Time ---*/
        delay(2); //delay for 2 * 5us = 10us
        //loop for pixels in sub-image row
        for (i=column - 1; i<end_column; i++)
        {
            data[0][i] = FPReadReg(FP_CTRLA);
            /*--- Wait A/D Conversion Time ---*/
            delay(1);
        }
    }
}

```

3.6 Sensor Register Map

Table 3 describes the MBF200 sensor register map and highlights the register offsets and certain important bit descriptions. Note that to read the value of register CTRLA, the shadow register SRA must be read. A direct read to CTRLA will return the A/D converter result.

Table 3. MBF200 Register Map

Register Offset	Register Name	Description	Contents
0x00	RAH	Row Address, High	Array Row Address
0x01	RAL	Row Address, Low	Array Row Address
0x02	CAL	Column Address, Low	Array Column Address
0x03	REH	Row Address End, High	Row Address for Sub-image
0x04	REL	Row Address End, Low	Row Address for Sub-image
0x05	CEL	Column Address End, Low	Column Address for Sub-image
0x06	DTR	Dichroic Time Register	Array Discharge Time
0x07	DCR	Discharge Current Register	Array Discharge Current
0x08	CTRLA	Control register A	Select Full, Sub-image or row capture
0x09	CTRLB	Control register B	Enable, A/D ready, Auto Finger Detect Enable, Auto-increment Enable
0x0A	CTRLC	Control register C	P0 and P1 Output Control
0x0B	SRA	Status Register A	Shadow of CTRLA
0x0C	PGC	Programmable Gain Control	Amplifier Gain Control
0x0D	ICR	Interrupt Control register	Interrupt Mask, Enable and Trigger Options
0x0E	ISR	Interrupt Status register	Interrupt Flags
0x0F	THR	Threshold Register	Auto Finger Detect Threshold
0x10	CIDH	Chip Identification, High	Chip ID, Always 0x20
0x11	CIDL	Chip Identification, Low	Chip ID
0x12	TST	Test Mode Register	Reserved

3.7 Sensor Conversion Parameters

The discharge time, discharge current, and gain values can be adjusted to suit the environment. However, it is necessary to first run the sensor and read from the array with no finger in order to get the ambient humidity readings. This enables appropriate values to be selected by comparing to readings when a finger is present. This ambient reading is also used by the polling loop which reads a row out of the sensor and checks the results against a value close to the ambient readings (in the example this value is 0xA0). If enough of the results are below this value, then a finger is present and the polling loop ends, returning to the main() function. The parameters can be changed using the FPAdjustParams() function shown below:

```

/*--- Adjust Parameters ---*/
void FPAdjustParams(uint8 discharge_time, uint8 discharge_current, uint8 gain)
{
    FPWriteReg(FP_DTR, discharge_time);
    FPWriteReg(FP_DCR, discharge_current);
    FPWriteReg(FP_PGC, gain);
}

```

3.8 Example Software main()

The main() function in the example software initializes the MCF5249 peripherals, LCD module, and fingerprint sensor before going into the startup display routines. The system then enters an endless loop in which it repeats the enrollment and verification sequences as requested by the user via the switch. For demonstration purposes in the reference design, the enrollment function is made available via this switch. In a standard system the enrollment function would be restricted to a master mode, to prevent unwanted users enrolling themselves.

The switch is tested to determine whether the user wants to go through the enrollment sequence or the verification sequence. Based on the result, the appropriate messages are displayed before the algorithm is run with specific options for enrollment or for verification. Note that the whole sensor image is extracted before the algorithm is run, and this image is provided as the input to the algorithm for both enrollment and verification.

The code below is taken from the example software and shows the main() function.

```

CPUInit();           //CPU module initializations
LCDInit();           //LCD initialization routine
FPSensorInit();     //Sensor initialization routine
LCDStartupDisplay(); //Runs the startup LCD sequence
while(1)
{
    func = (MCF5249_SIM_GPIO1_READ & 0x00100000);
    if (func != 0) //Test the switch for enrollment or verification
        LCDEnrollement(); //Start of enrollment display
    else
        LCDVerification(); //Start of verification display
    FPMCUFingerDetect(); //Only returns when a finger is detected
    FPMCUGetWhole(image1); //Get the whole image from sensor
    if (func !=0)
    {
        Enrollement_Algorithm(image1); //Run the algorithm
        LCDUserEnrolled(); //Enrollment completed display
        while(func !=0)
            func = (MCF5249_SIM_GPIO1_READ & 0x00100000); //Test the switch
    }
    else

```

Freescale Semiconductor, Inc.

Summary

```
{  
    if(Verification_Algorithm(image1)) //Run the algorithm  
        LCDUserVerified(); //User verified display  
    else  
        LCDUsedRejected(); //User rejected display  
}
```

4 Summary

This application note has detailed the hardware design and software development of the MCF5249 Biometrics reference design, demonstrating the ability of the MCF5249 to perform excellently in a biometric fingerprint application. Design schematics and a BOM for the hardware, application drivers and example software, and additional reference material are available from the MCF5249 Biometrics web page to complement this application note.

THIS PAGE INTENTIONALLY LEFT BLANK

THIS PAGE INTENTIONALLY LEFT BLANK

Freescale Semiconductor, Inc.

HOW TO REACH US:

USA/EUROPE/LOCATIONS NOT LISTED:

Motorola Literature Distribution
P.O. Box 5405, Denver, Colorado 80217
1-303-675-2140 or 1-800-441-2447

JAPAN:

Motorola Japan Ltd.
SPS, Technical Information Center
3-20-1, Minami-Azabu Minato-ku
Tokyo 106-8573 Japan
81-3-3440-3569

ASIA/PACIFIC:

Motorola Semiconductors H.K. Ltd.
Silicon Harbour Centre, 2 Dai King Street
Tai Po Industrial Estate, Tai Po, N.T., Hong Kong
852-26668334

TECHNICAL INFORMATION CENTER:

1-800-521-6274

HOME PAGE:

<http://www.motorola.com/semiconductors>

Information in this document is provided solely to enable system and software implementers to use Motorola products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Motorola reserves the right to make changes without further notice to any products herein.

Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Motorola data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part.



Motorola and the Stylized M Logo are registered in the U.S. Patent and Trademark Office. digital dna is a trademark of Motorola, Inc. All other product or service names are the property of their respective owners. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

© Motorola, Inc. 2002

AN2382/D

**For More Information On This Product,
Go to: www.freescale.com**