Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	SAM8RC
Core Size	8-Bit
Speed	12MHz
Connectivity	I²C, UART/USART
Peripherals	LCD, LVD, LVR, PWM, WDT
Number of I/O	22
Program Memory Size	8KB (8K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	272 x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 5.5V
Data Converters	A/D 13x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Through Hole
Package / Case	20-DIP (0.300", 7.62mm)
Supplier Device Package	20-DIP
Purchase URL	https://www.e-xfl.com/product-detail/zilog/s3f8s28xzz-dk98

4 Control Registers 4-1

4.1 Overview 4-1
4.1.1 ADCON 4-6
4.1.2 BTCON 4-7
4.1.3 CLKCON 4-8
4.1.4 EMT 4-9
4.1.5 FLAGS 4-10
4.1.6 FMCON 4-11
4.1.7 FMSECH 4-11
4.1.8 FMSECL 4-12
4.1.9 FMUSR 4-12
4.1.10 ICCR 4-13
4.1.11 ICSR 4-14
4.1.12 IMR 4-15
4.1.13 IPH 4-16
4.1.14 IPL 4-16
4.1.15 IPR 4-17
4.1.16 IRQ 4-18
4.1.17 LVDCON 4-19
4.1.18 P0CONH 4-20
4.1.19 P0CONL 4-21
4.1.20 P0PND 4-22
4.1.21 P0PUR 4-24
4.1.22 P1CON 4-25
4.1.23 P2CONH 4-26
4.1.24 P2CONL 4-27
4.1.25 P2PUR 4-28
4.1.26 P3CON 4-29
4.1.27 P3PND 4-30
4.1.28 PP 4-31
4.1.29 PWM0CON 4-32
4.1.30 PWM1CON 4-33
4.1.31 PWM0EX 4-34
4.1.32 PWM1EX 4-34
4.1.33 RESETID 4-35
4.1.34 ROSCCON 4-36
4.1.35 RP0 4-36
4.1.36 RP1 4-37
4.1.37 SPL 4-37
4.1.38 STOPCON 4-37
4.1.39 SYM 4-38
4.1.40 T1CON 4-39
4.1.41 T1PS 4-40
4.1.42 TACON 4-40
4.1.43 TBCON 4-42
4.1.44 UARTCON 4-43
4.1.45 UARTPND 4-44
4.1.46 WDTCON 4-45

5 Interrupt Structure 5-1

5.1 Overview 5-1
5.1.1 Levels 5-1
5.1.2 Vectors 5-1
5.1.3 Sources 5-1

5.2 Interrupt Types 5-2
5.3 S3F8S28/S3F8S24 Interrupt Structure 5-3
5.3.1 Interrupt Vector Addresses 5-4
5.3.2 Enable/Disable Interrupt Instructions (EI, DI) 5-4
5.4 System-Level Interrupt Control Registers 5-5
5.5 Interrupt Processing Control Points 5-6
5.6 Peripheral Interrupt Control Registers 5-7
5.7 System Mode Register (SYM) 5-8
5.8 Interrupt Mask Register (IMR) 5-9
5.9 Interrupt Priority Register (IPR) 5-10
5.10 Interrupt Request Register (IRQ) 5-12
5.11 Interrupt Pending Function Types 5-13
5.11.1 Overview 5-13
5.11.2 Pending Bits Cleared Automatically by Hardware 5-13
5.11.3 Pending Bits Cleared by the Service Routine 5-13
5.12 Interrupt Source Polling Sequence 5-14
5.13 Interrupt Service Routines 5-14
5.14 Generating Interrupt Vector Addresses 5-15
5.15 Nesting of Vectored Interrupts 5-15
5.16 Instruction Pointer (IP) 5-15
5.17 Fast Interrupt Processing 5-16
5.18 Procedure for Initiating Fast Interrupts 5-16
5.19 Fast Interrupt Service Routine 5-16
5.20 Relationship to Interrupt Pending Bit Types 5-17
5.21 Programming Guidelines 5-17

6 Instruction Set 6-1

6.1 Overview 6-1
6.1.1 Data Types 6-1
6.1.2 Register Addressing 6-1
6.1.3 Addressing Modes 6-1
6.2 Flags Register (FLAGS) 6-5
6.2.1 Flag Descriptions 6-6
6.3 Instruction Set Notation 6-7
6.4 Condition Codes 6-11
6.5 Instruction Descriptions 6-12
6.5.1 ADC (Add with Carry) 6-13
6.5.2 ADD (Add) 6-14
6.5.3 AND (Logical AND) 6-15
6.5.4 BAND (Bit AND) 6-16
6.5.5 BCP (Bit Compare) 6-17
6.5.6 BITC (Bit Complement) 6-18
6.5.7 BITR (Bit Reset) 6-19
6.5.8 BITS (Bit Set) 6-20
6.5.9 BOR (Bit OR) 6-21
6.5.10 BTJRF (Bit Test, Jump Relative on False) 6-22
6.5.11 BTJRT (Bit Test, Jump Relative on True) 6-23
6.5.12 BXOR (Bit XOR) 6-24
6.5.13 CALL (Call Procedure) 6-25
6.5.14 CCF (Complement Carry Flag) 6-26
6.5.15 CLR (Clear) 6-27
6.5.16 COM (Complement) 6-28
6.5.17 CP (Compare) 6-29
6.5.18 CPIJE (Compare, Increment, and Jump on Equal) 6-30

16.7 Read/Write Operations 16-14
16.8 Bus Arbitration Procedures 16-14
16.9 Abort Conditions 16-14
16.10 Configuring the IIC-Bus 16-14

17 Low Voltage Detector 17-1

17.1 Overview 17-1
17.2 Low Voltage Detector Control Register (LVDCON) 17-2
17.3 Voltage (VDD) Level Detection Sequence-LVD Usage 17-4

18 Embedded Flash Memory Interface 18-1

18.1 Overview 18-1
18.1.1 Flash ROM Configuration 18-1
18.1.2 Tool Program Mode 18-2
18.1.3 User Program Mode 18-2
18.2 Flash Memory Control Registers (User Program Mode) 18-3
18.2.1 Flash Memory Control Register (FMCON) 18-3
18.2.2 Flash Memory User Programming Enable Register (FMUSR) 18-3
18.2.3 Flash Memory Sector Address Registers 18-4
18.3 ISP™ (On-Board Programming) Sector 18-5
18.3.1 ISP Reset Vector and ISP Sector Size 18-6
18.4 Sector Erase 18-7
18.5 Programming 18-9
18.6 Reading 18-14
18.7 Hard Lock Protection 18-15

19 Electrical Data 19-1

19.1 Overview 19-1

20 Mechanical Data 20-1

20.1 Overview 20-1

21 Flash MCU 21-1

21.1 Overview 21-1
21.2 On Board Writing 21-4

22 Development Tools 22-6

22.1 Overview 22-6
22.2 Emulator-based Development System 22-6
22.2.1 Host Software 22-7
22.2.2 Target Boards 22-7
22.2.3 SMDS2+ Selection (SAM8) 22-10
22.3 Zilog Library-based Development Platform 22-14
22.3.1 Zilog Developer Platform Components 22-14
22.3.2 Compatibility with 3rd Party Tools 22-16
22.3.3 Benefits and Limitations of Zilog Development Tools 22-16
22.3.4 Development Tools 22-16

3.5 Direct Address Mode (DA)

In Direct Address (DA) mode, the instruction provides the operand's 16-bit memory address. Jump (JP) and Call (CALL) instructions use this addressing mode to specify the 16-bit destination address that is loaded into the PC whenever a JP or CALL instruction is executed.

The LDC and LDE instructions can use Direct Address mode to specify the source or destination address for Load operations to program memory (LDC) or to external data memory (LDE), if implemented.

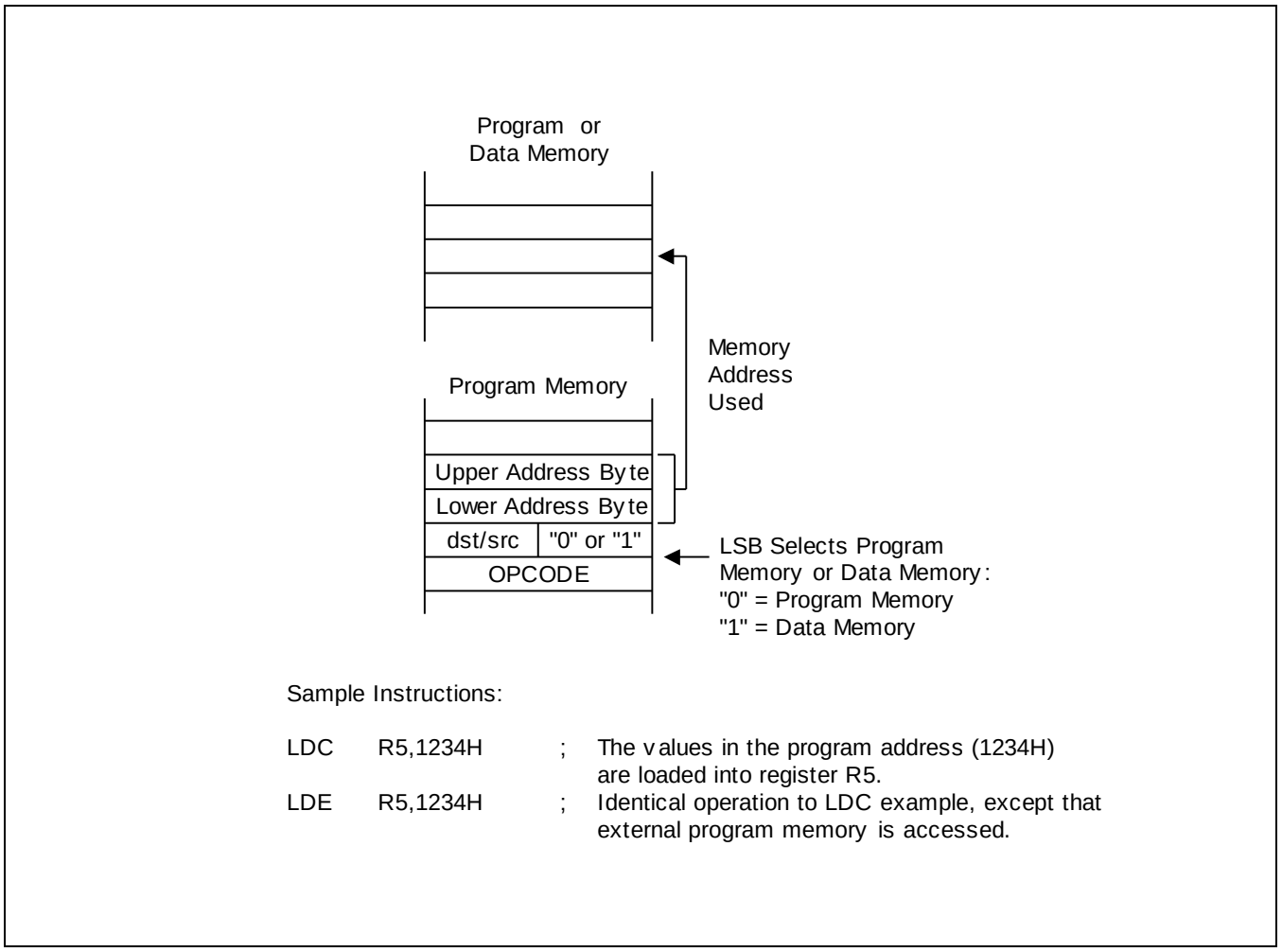


Figure 3-10 Direct Addressing for Load Instructions

4.1.8 FMSECL

- Flash Memory Sector Address Register (Low Byte): EFH, SET 1, BANK 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
Reset Value	0	0	0	0	0	0	0	0
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW

.7 Flash Memory Sector Address Bit (Low Byte)

The 7th bit to select a sector of Flash ROM

.6–.0 Bits 6–0

Don't care

NOTE: The low-byte Flash memory sector address pointer value is the lower eight bits of the 16-bit pointer address.

4.1.9 FMUSR

- Flash Memory User Programming Enable Register: EDH, SET 1, BANK 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
Reset Value	0	0	0	0	0	0	0	0
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW

.7–.0 Flash Memory User Programming Enable Bits

10100101	Enable user programming mode
Other values	Disable user programming mode

4.1.19 P0CONL

- Port 0 Control Register (Low Byte): E7H, SET 1, BANK 0

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
Reset Value	0	0	0	0	0	0	0	0
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW

.7–.6 Port 0, P0.3/INT3 Configuration Bits

0	0	Schmitt trigger input/falling edge interrupt input
0	1	Alternative function: SDA input
1	0	Push-pull output
1	1	A/D converter input (ADC3); Schmitt trigger input off

.5–.4 Port 0, P0.2/ADC2 Configuration Bits

0	0	Schmitt trigger input/falling edge interrupt input
0	1	Alternative function: SCK input
1	0	Push-pull output
1	1	A/D converter input (ADC2); Schmitt trigger input off

.3–.2 Port 0, P0.1/ADC1/INT1 Configuration Bits

0	x	Schmitt trigger input/falling edge interrupt input
1	0	Push-pull output
1	1	A/D converter input (ADC1); Schmitt trigger input off

.1–.0 Port 0, P0.0/ADC0/INT0 Configuration Bits

0	x	Schmitt trigger input/falling edge interrupt input
1	0	Push-pull output
1	1	A/D converter input (ADC0); Schmitt trigger input off

4.1.24 P2CONL

- Port 2 Control Register (Low Byte): EBH, SET 1, BANK 0

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
Reset Value	0	0	0	0	0	0	0	0
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW

.7–.6 Part 2, P2.3 Configuration Bits

0	0	Schmitt trigger input
0	1	Alternative function: TxD output
1	0	Push-pull output
1	1	Open-drain output

.5–.4 Port 2, P2.2 Configuration Bits

0	0	Schmitt trigger input T1 capture input; RxD input
0	1	Alternative function: RxD output
1	0	Push-pull output
1	1	Open-drain output

.3–.2 Port 2, P2.1 Configuration Bits

0	0	Schmitt trigger input
0	1	Alternative function:T1 match output
1	0	Push-pull output
1	1	Open-drain output

.1–.0 Port 2, P2.0 Configuration Bits

0	0	Schmitt trigger input
0	1	Alternative function:T0 match output
1	0	Push-pull output
1	1	Open-drain output

.0 Port 3.0/ADC9/INT4 Interrupt Pending Bit

0	No interrupt pending (when read)
0	Pending bit clear (when write)
1	Interrupt pending (when read)
1	No effect (when write)

4.1.28 PP

- Register Page Pointer: DFH, SET 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
Reset Value	0	0	0	0	0	0	0	0
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW

.7–.0	Not used for the S3F8S28/S3F8S24.

NOTE: In S3F8S28/S3F8S24, only page 0 settings are valid. Register page pointer values for the source and destination register page are automatically set to "00F" following a hardware reset. These values should not be changed during normal operation.

5.14 Generating Interrupt Vector Addresses

The interrupt vector area in the ROM (00H to FFH) contains the addresses of interrupt service routines that correspond to each level in the interrupt structure. Vectored interrupt processing follows this sequence:

1. Push the program counter's low-byte value to the stack.
2. Push the program counter's high-byte value to the stack.
3. Push the FLAG register values to the stack.
4. Fetch the service routine's high-byte address from the vector location.
5. Fetch the service routine's low-byte address from the vector location.
6. Branch to the service routine specified by the concatenated 16-bit vector address.

NOTE: A 16-bit vector address always begins at an even-numbered ROM address within the range of 00H to FFH.

5.15 Nesting of Vectored Interrupts

It is possible to nest a higher-priority interrupt request while a lower-priority request is being serviced. To do this, you must follow these steps:

1. Push the current 8-bit interrupt mask register (IMR) value to the stack (PUSH IMR).
2. Load the IMR register with a new mask value that enables only the higher priority interrupt.
3. Execute an EI instruction to enable interrupt processing (a higher priority interrupt will be processed if it occurs).
4. When the lower-priority interrupt service routine ends, execute DI, restore the IMR to its original value by returning the previous mask value from the stack (POP IMR).
5. Execute an IRET.

Depending on the application, you may be able to simplify the procedure above to some extent.

5.16 Instruction Pointer (IP)

The instruction pointer (IP) is adopted by all the S3C8/S3F8 Series microcontrollers to control the optional high-speed interrupt processing feature called fast interrupts. The IP consists of register pair DAH and DBH. The names of IP registers are IPH (high byte, IP15 to IP8) and IPL (low byte, IP7 to IP0).

6.5.12 BXOR (Bit XOR)

BXOR dst, src.b

BXOR dst.b, src

Operation: dst (0) ← dst (0) XOR src (b)
or
dst (b) ← dst (b) XOR src (0)
The specified bit of the source (or the destination) is logically exclusive-ORed with bit zero (LSB) of the destination (or source). The result bit is stored in the specified bit of the destination. No other bits of the destination are affected. The source is unaffected.

Flags: **C**: Unaffected.
Z: Set if the result is "0"; cleared otherwise.
S: Cleared to "0".
V: Undefined.
D: Unaffected.
H: Unaffected.

Format:

			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc	dst | b | 0	src	3	6	27	r0	Rb
opc	src | b | 1	dst	3	6	27	Rb	r0

NOTE: In the second byte of the 3byte instruction formats, the destination (or source) address is four bits, the bit address "b" is three bits, and the LSB address value is one bit in length.

Examples: Given: R1 = 07H (00000111B) and register 01H = 03H (00000011B):

BXOR R1, 01H.1 → R1 = 06H, Register 01H = 03H
BXOR 01H.2, R1 → Register 01H = 07H, R1 = 07H

In the first example, destination working register R1 has the value 07H (00000111B) and source register 01H has the value 03H (00000011B). The statement "BXOR R1, 01H.1" exclusive-ORs bit one of register 01H (source) with bit zero of R1 (destination). The result bit value is stored in bit zero of R1, changing its value from 07H to 06H. The value of source register 01H is unaffected.

6.5.29 IDLE (Idle Operation)

IDLE

Operation: The IDLE instruction stops the CPU clock while allowing system clock oscillation to continue. Idle mode can be released by an interrupt request (IRQ) or an external reset operation.
In application programs, a IDLE instruction must be immediately followed by at least three NOP instructions. This ensures an adeguate time interval for the clock to stabilize before the next instruction is executed. If three or more NOP instructons are not used after IDLE instruction, leakage current could be flown because of the floating state in the internal bus.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	src
opc	1	4	6F	–	–

Example: The instruction

```
        IDLE            ; stops the CPU clock but not the system clock
NOP
NOP
NOP
```

Example 8-1 Sample S3F8S28/S3F8S24 Initialization Routine

```
;--------------<< Interrupt Vector Address >>
              ORG     0000H
              VECTOR  0F2H, PWM0OVF_INT     ;
              VECTOR  0F4H, INT_TIMERB      ;
              VECTOR  0F6H, INT_TIMERA      ;
              VECTOR  0FCH, INT_EXT1        ;
              VECTOR  0FEH, INT_EXT0        ;

;--------------<< Smart Option >>
              ORG     003CH
              DB      0FFH                  ; 003CH, must be initialized to 0
              DB      0FFH                  ; 003DH, must be initialized to 0
              DB      0FFH                  ; 003EH, enable LVR
              DB      0FEH                  ; 003FH, External RC oscillator

;--------------<< Initialize System and Peripherals >>
              ORG     0100H
RESET:        DI                            ; Disable interrupt
              LD      BTCON,#10100011B      ; Watch-dog disable
              LD      CLKCON,#00011000B     ; Select non-divided CPU clock
              LD      SP,#0C0H              ; Stack pointer must be set

              LD      P0CONH,#10101010B     ;
              LD      P0CONL,#10101010B     ; P0.0-P0.7 push-pull output
              LD      P0PND,#00001010B      ; P0.0, P0.1 interrupt enable
              LD      P1CON,#00001000B      ; P1.1 push-pull output
              LD      P2CONH,#01001010B     ;
              LD      P2CONL,#10101010B     ; P2.0-P2.6 push-pull output

              LD      IMR,#00000111B        ; Enable IRQ0, IRQ1, IRQ2 interrupt
              LD      IPR,#00010011B        ; IRQ2>IRQ1>IRQ0

;--------------<< Timer 0 settings >>
              LD      TADATA,#50H           ; CPU = 4MHz, interrupt interval = 6.4msec
              LD      TBDATA,#50H
              LD      TACON,#00000110B      ; fOSC/256, Timer A interrupt enable
              LD      TBCON,#00000110B      ; fOSC/256, Timer B interrupt enable

;--------------<< Initialize other registers >>
              •
              •
              EI                            ; Enable interrupt

;--------------<< Main loop >>
MAIN:         NOP                           ; Start main loop
              LD      BTCON,#02H            ; Enable watchdog function
                                            ; Basic counter (BTCNT) clear
              •
              •
              CALL    KEY_SCAN              ;
              •
```

12.6 Interrupt & System Reset

If all the watchdog interrupt and overflow reset are enabled, and when you enable the watchdog, the counter start to counting, the interrupt will be generated at the counter reach to 0x3FF, and the counter continuous to counting, if the counter overflow, the overflow reset will generated.

This operating mechanism combines the two events by first giving an interrupt and then giving a reset. This will for instance allow a safe shutdown by saving critical parameters before a system reset.

There is a possibility to set a pending window where users can restart the watchdog counter within this window. When the interrupt occurred, user can clear the counter to prevent the internal reset.

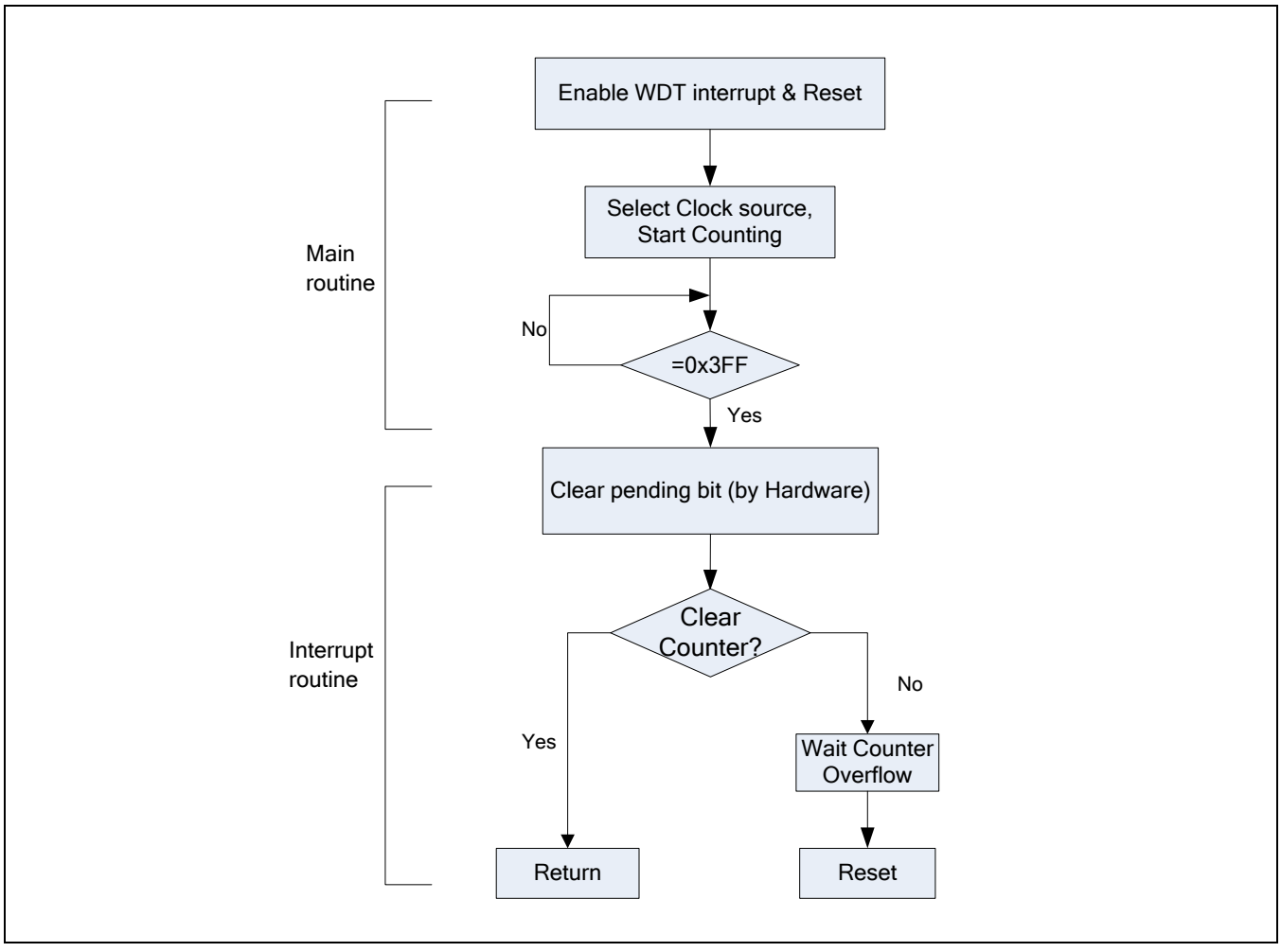


Figure 12-5 Interrupt & System Reset Operation Sequence

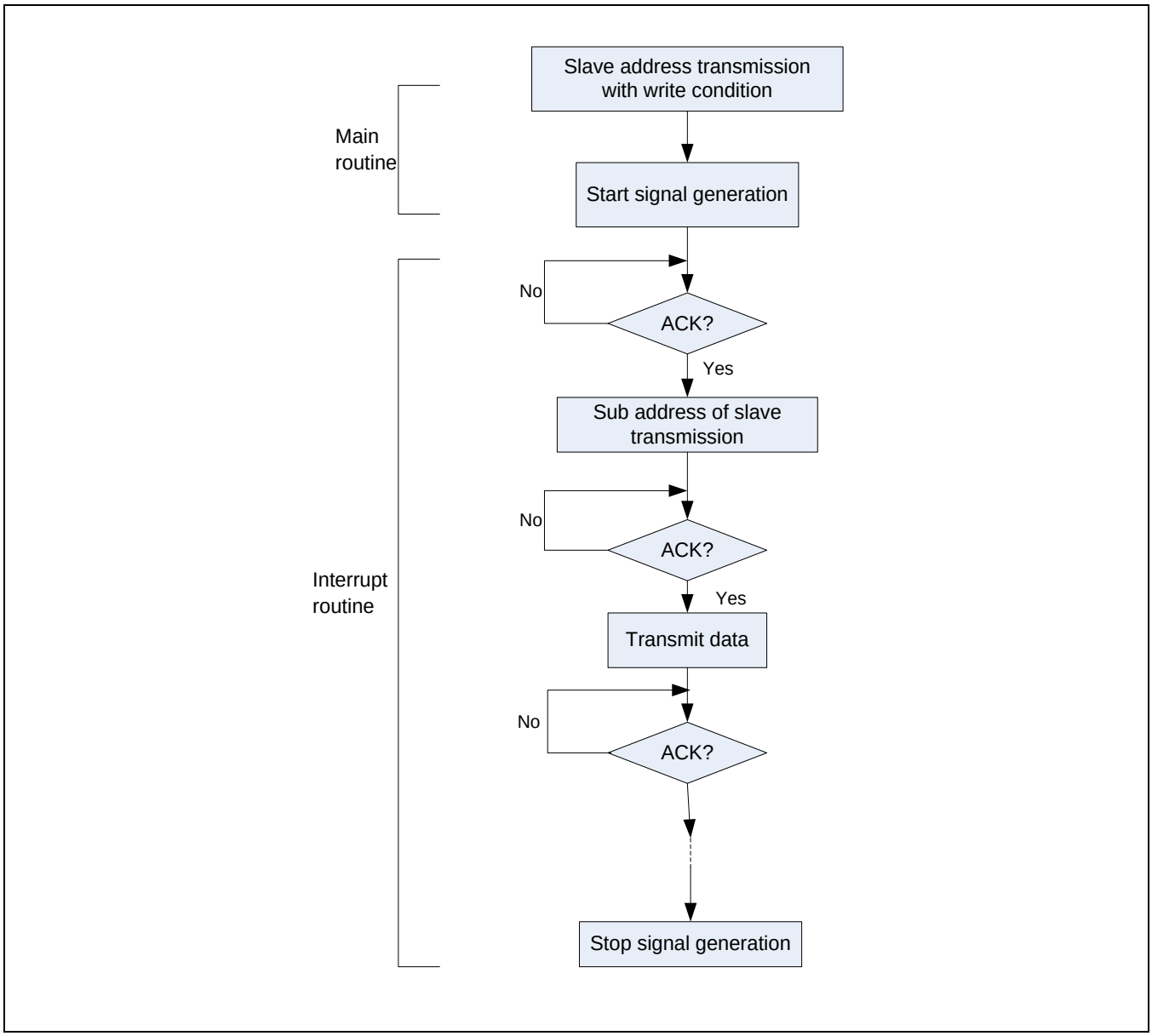


Figure 16-11 Write Operation Sequence

17 Low Voltage Detector

17.1 Overview

The S3F8S28/S3F8S24 micro-controller has a built-in LVD (Low Voltage Detector) circuit which allows detection of power voltage drop to generate flag:

- Generate flag when VDD less than one selected level from 4.1, 3.2, 2.5 or 2.1V

Low voltage detector circuits have following functional components:

- Enable or disable LVD module
- LVD Flag when detector setting level.

Turning the LVD operation on and off can be controlled by software. Because the IC consumes a large amount of current during LVD operation, it is recommended that the LVD operation should be kept OFF unless it is necessary.

Also the LVD criteria voltage can be set by the software. The LVD flag criteria voltage can be set by matching to one of the 4 kinds of voltage 2.1V, 2.5V, 3.2V,4.1V (VDD reference voltage).

The LVD block works only when LVDCON.7 is set. If VDD level is lower than the reference voltage selected with LVDCON.1-0, LVDCON.5 will be set. If VDD level is higher, LVDCON.5 will be cleared.

Table 19-5 Oscillation Stabilization Time

(T_A = – 40°C to + 85°C, V_{DD} = 1.8V to 5.5V)

Oscillator	Test Condition	Min.	Typ.	Max.	Unit
Main crystal	f_{OSC} > 1.0MHz	–	–	20	ms
Main ceramic	Oscillation stabilization occurs when V_{DD} is equal to the minimum oscillator voltage range.	–	–	10	ms
External clock (main system)	X_{IN} input high and low width (t_{XH}, t_{XL})	25	–	500	ns
Oscillator stabilization wait time	t_{WAIT} when released by a reset (1)	–	$2^{19}/f_{OSC}$	–	ms
	t_{WAIT} when released by an interrupt (2)	–	–	–	ms

NOTE:

1. f_{OSC} is the oscillator frequency.
2. The duration of the oscillator stabilization wait time, t_{WAIT}, when it is released by an interrupt is determined by the settings in the basic timer control register, BTCON.

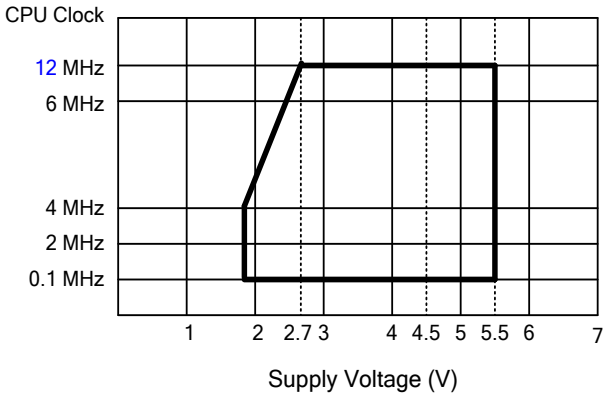
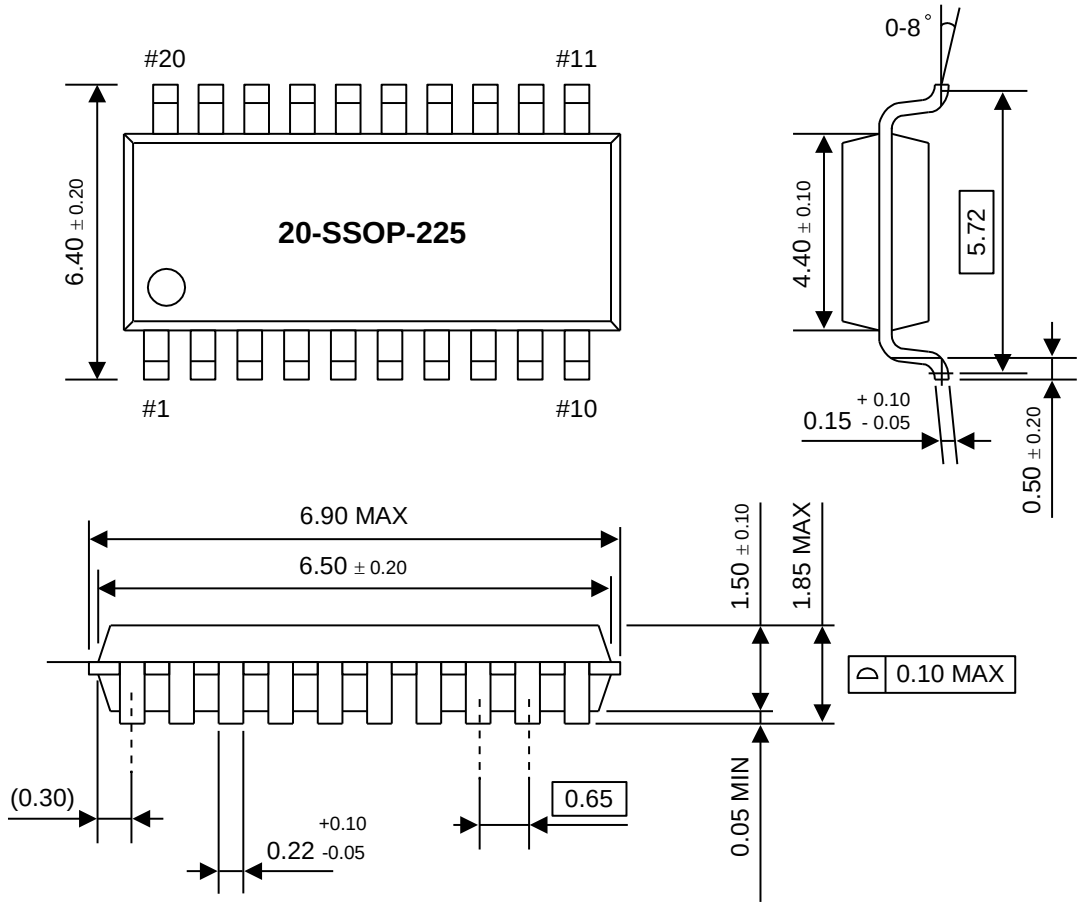


Figure 19-2 Operating Voltage Range



NOTE: Dimensions are in millimeters.

Figure 20-5 20-SSOP-225 Package Dimensions

21.2 On Board Writing

The S3F8S28/S3F8S24 needs only 5 signal lines including V_{DD} and GND pins for writing internal Flash memory with serial protocol. Therefore the on-board writing is possible if the writing signal lines are considered when the PCB of application board is designed.

Circuit Design Guide:

At the Flash writing, the writing tool needs 5 signal lines that are GND, V_{DD}, V_{PP}, SDAT and SCLK. When you design the PCB circuits, you should consider the usage of these signal lines for the on-board writing.

In case of V_{PP} (nRESET) pin, for the purpose of increase the noise effect, a capacitor should be inserted between the V_{PP} pin and GND.

Please be careful to design the related circuit of these signal pins because rising/falling timing of V_{PP}, SCLK and SDAT is very important for proper programming.

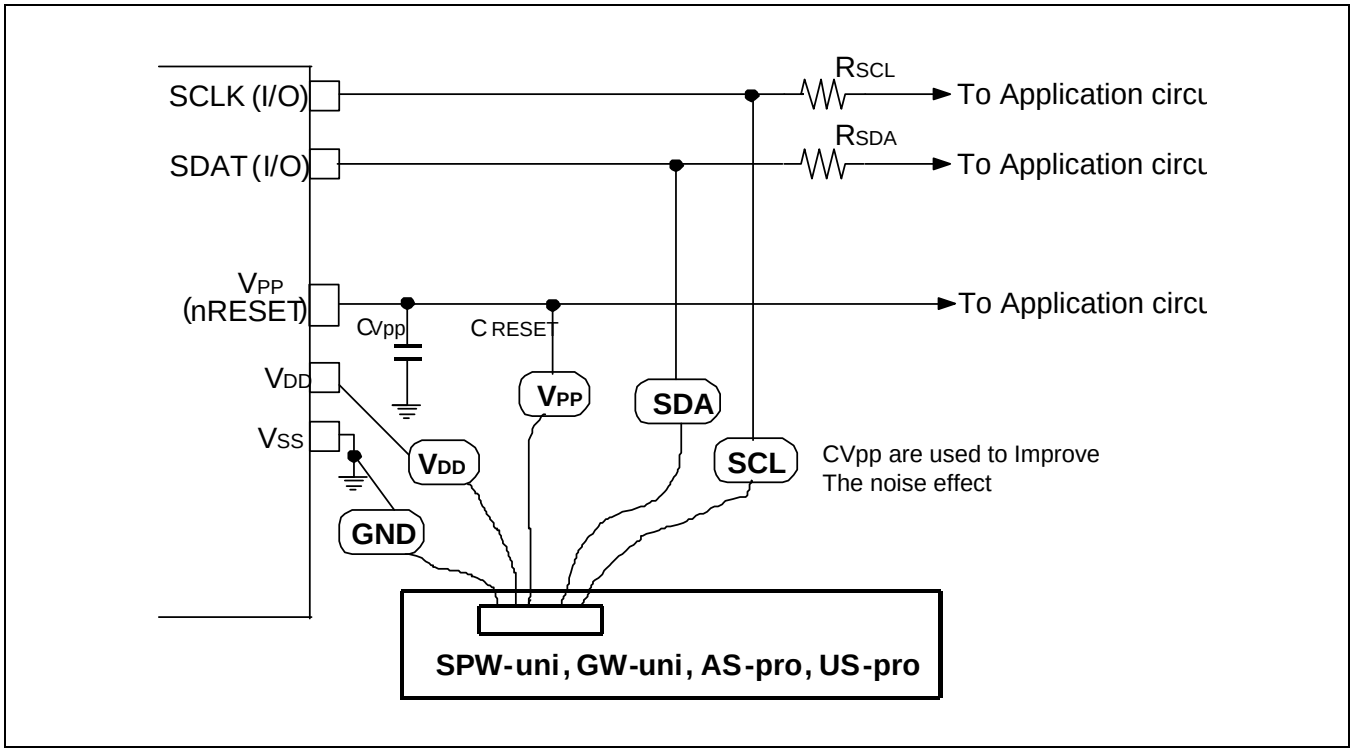


Figure 21-3 PCB Design Guide for on Board Programming

The S3 Emulator Based Development System includes the components listed in the following sections.

22.2.1 Host Software

Host software is required to create and debug S3 application programs in C or assembly language. The host software program converts the application source code into an executable format that is downloaded into the evaluation (EVA) chip on the target board for program execution/debugging. Optionally, the probe adapter cable(s) can be connected between the target board and the application board to debug program interaction with components on the application board.

Zilog provides the Zilog Developer Studio (ZDS) software suite host software package free of charge for any PC running a supported version of the Windows operating system. Alternatively, 3rd party host software packages (such as the IAR Embedded Workbench host software package) are available for purchase from vendor websites. The ZDS S3 software package is available for free download from the Zilog website.

22.2.2 Target Boards

Target boards are available for all S3C8/S3F8-series microcontrollers. Each target board includes the cables and adapters necessary to interface with an application board. The target board can be used with a 3rd party emulator to enable application debugging with or without an application board. Alternatively, the emulator can be used to program the target MCU on the application board using the supplied 10- circuit programming cable. The TB8S19/8S28/8S19 target board can be used with application boards targeting the S3F8S19, S3F8S28, and S3F8S39 MCUs.

Figure 22-2 shows how the TB8S19/8S28/8S19 Target Board is configured. The symbol “ ” marks the starting point of the jumper signals.

	**S3 Flash In-System Programmer II** Zilog's S3 Flash ISP II provides an interface between any development or application board with an S3 microcontroller device to the high-speed USB port of a PC on which Zilog Developer Studio II for S3 Family devices (ZDS II – S3) is installed. The ISP II allows the Flash memory space on any S3 Family device to be programmed, and also offers limited debugging capabilities when used together with the Zilog Debug Library. The following features are available with the S3 Flash ISP II when using ZDS II for S3 Family devices: • Download code to Flash and begin to program execution • Break program execution arbitrarily • Single-step debugging of the application, view/edit memory and S3 special function registers. Resume normal program operation after a breakpoint • Insert multiple breakpoints in a program at compile/assembly time	Zilog • TEL: (408) 457-9000 • FAX: (408) 416-0223 • E-mail: s3sales@zilog.com • URL: *http://www.zilog.com*

To obtain the S3 Family development tools that will satisfy your S3F8S28/S3F8S24 development objectives, contact your local Zilog Sales Office, or visit Zilog's Third Party Tools page to review our list of third party tool suppliers.