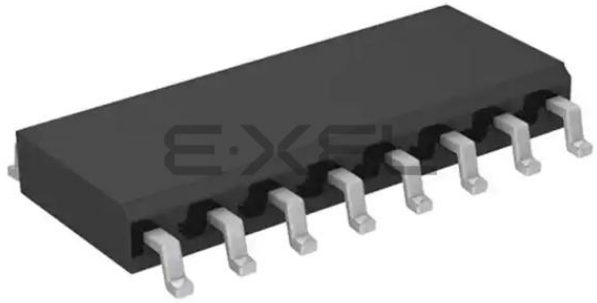


Welcome to E-XFL.COM

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	8051
Core Size	8-Bit
Speed	24MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, KPI, LCD, POR, PWM, WDT
Number of I/O	25
Program Memory Size	16KB (16K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	2.4V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	16-SOIC (0.154", 3.90mm Width)
Supplier Device Package	16-SOP
Purchase URL	https://www.e-xfl.com/product-detail/nuvoton-technology-corporation-america/n79e715as16

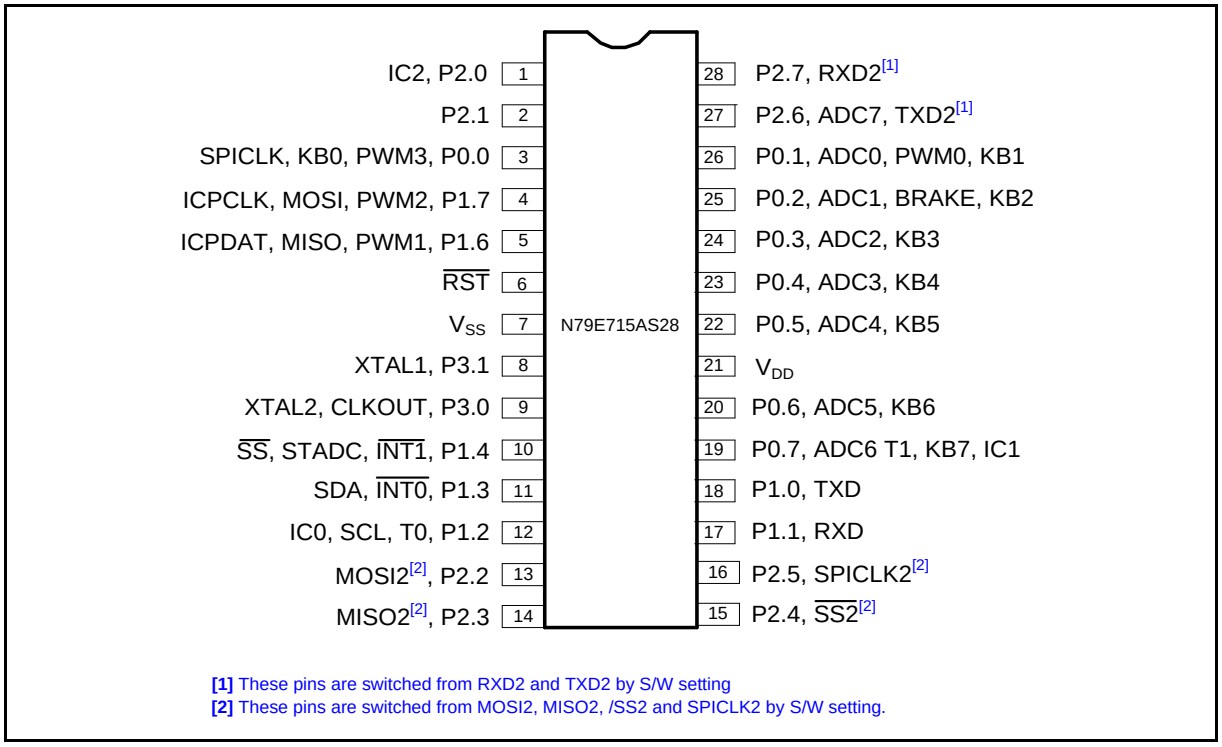


Figure 5-3 SOP 28-pin Assignment

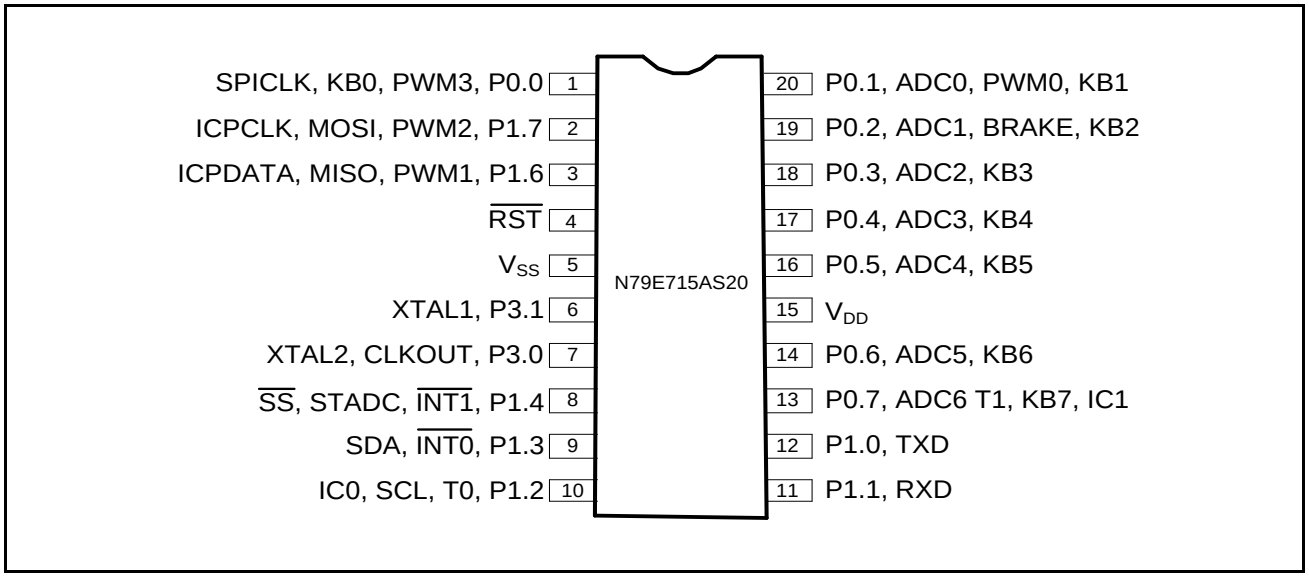


Figure 5-4 SOP 20-pin Assignment

Bit	Name	Description
5	F0	**User Flag 0** The general-purpose flag that can be set or cleared by the user.
4	RS1	**Register Bank Selecting Bits** **The two bits select one of four banks in which R0~R7 locate.** RS1 RS0 Register Bank RAM Address 0 0 0 00~07H 0 1 1 08~0FH 1 0 2 10~17H 1 1 3 18~1FH
3	RS0	
2	OV	**Overflow Flag** OV is used for a signed character operands. For an ADD or ADDC instruction, OV will be set if there is a carry out of bit 6 but not out of bit 7, or a carry out of bit 7 but not bit 6. Otherwise, OV is cleared. OV indicates a negative number produced as the sum of two positive operands or a positive sum from two negative operands. For a SUBB, OV is set if a borrow is needed into bit6 but not into bit 7, or into bit7 but not bit 6. Otherwise, OV is cleared. OV indicates a negative number produced when a negative value is subtracted from a positive value or a positive result when a positive number is subtracted from a negative number. For a MUL, if the product is greater than 255 (00FFH), OV will be set. Otherwise, it is cleared. For a DIV, it is normally 0. However, if B had originally contained 00H, the values returned in A and B will be undefined. Meanwhile, the OV will be set.
1	F1	**User Flag 1** The general purpose flag that can be set or cleared by the user via software.
0	P	**Parity Flag** Set to 1 to indicate an odd number of ones in the accumulator. Cleared for an even number of ones. It performs even parity check.

Table 8–1 Instructions that Affect Flag Settings

Instruction	CY	OV	AC	Instruction	CY	OV	AC
ADD	X[1]	X	X	CLR C	0		
ADDC	X	X	X	CPL C	X		
SUBB	X	X	X	ANL C, bit	X		
MUL	0	X		ANL C, /bit	X		
DIV	0	X		ORL C, bit	X		
DA A	X			ORL C, /bit	X		
RRC A	X			MOV C, bit	X		

Bit	Name	Description
4	P0S	Enable Schmitt trigger inputs on Port 0.
1	P3M1.1	Control the output configuration of P3.1.
0	P3M1.0	Control the output configuration of P3.0.

P3M2 – Port3 Output Mode2

7	6	5	4	3	2	1	0
-	-	-	-	-	ENCLK	P3M2.1	P3M2.0
-	-	-	-	-	R/W	R/W	R/W

Address: 97H Reset value: 0000 0000B

Bit	Name	Description
7:3	-	**Reserved**
0	ENCLK	Clock Output to XTAL2 Pin (P3.0) Enable If the clock is from HIRC, the frequency of P3.0 is $F_{HIRC}/4$.
1	P3M2.1	Refer to Table 9-1 Setting Table for I/O Port Structure
0	P3M2.0	

Bit	Name	Description
6:4	T2DIV[2:0]	**Timer 2 Clock Divider** 000 = Timer 2 clock divider is 1/4. 001 = Timer 2 clock divider is 1/8. 010 = Timer 2 clock divider is 1/16. 011 = Timer 2 clock divider is 1/32. 100 = Timer 2 clock divider is 1/64. 101 = Timer 2 clock divider is 1/128. 110 = Timer 2 clock divider is 1/256. 111 = Timer 2 clock divider is 1/512.
3	CAPCR	**Capture Auto-clear** This bit enables auto-clear Timer 2 value in TH2 and TL2 when a determined input capture event occurs. 0 = Timer 2 continues counting when a capture event occurs. 1 = Timer 2 value is auto-cleared as 0000H when a capture event occurs.
2	COMPCR	**Compare Match Auto-clear** This bit enables auto-clear Timer 2 value in TH2 and TL2 when a compare match occurs. 0 = Timer 2 continues counting when a compare match occurs. 1 = Timer 2 value is auto-cleared as 0000H when a compare match occurs.
1:0	LDTS[1:0]	**Auto-reload Trigger Selection** These bits select the reload trigger event. 00 = Reload when Timer 2 overflows. 01 = Reload when input capture 0 event occurs. 10 = Reload when input capture 1 event occurs. 11 = Reload when input capture 2 event occurs.

RCOMP2L – Timer 2 Reload/Compare Low Byte

7	6	5	4	3	2	1	0
RCOMP2L[7:0]							
R/W							

Address: CAH Reset value: 0000 0000B

Bit	Name	Description
7:0	RCOMP2L[7:0]	**Timer 2 Reload/Compare Low Byte** This register stores the low byte of compare value when Timer 2 is configured in compare mode, It holds the low byte of the reload value when auto-reload mode.

RCOMP2H – Timer 2 Reload/Compare High Byte

7	6	5	4	3	2	1	0
RCOMP2H[7:0]							
R/W							

Address: CBH Reset value: 0000 0000B

Bit	Name	Description
7:0	RCOMP2H[7:0]	**Timer 2 Reload/Compare High Byte** This register stores the high byte of compare value when Timer 2 is configured in compare mode. Also, it holds the high byte of the reload value when auto-reload mode.

EIE – Extensive Interrupt Enable

7	6	5	4	3	2	1	0
ET2	ESPI	EPWM	EWDI	-	ECPTF	EKB	EI2C
R/W	R/W	R/W	R/W	-	R/W	R/W	R/W

Address: E8H Reset value: 0000 0000B

Bit	Name	Description
4	EWDI	0 = Disable Watchdog Timer Interrupt. 1 = Enable Watchdog Timer Interrupt.

The Watchdog Timer time-out selection will result in different time-out values depending on the clock speed. The reset, when enabled, will occur when 512 clocks after time-out has occurred.

Table 11-1 Time-out Values for the Watchdog Timer

(WPS2,WPS1,WPS0)	Pre-scalar	WDT Interrupt time-out		Reset time-out	
		Number of Clocks	Time	Number of Clocks	Time
(0,0,0)	1/1	2^6	6.4ms	2^6+512	57.6ms
(0,0,1)	1/2	2x2^6	12.8ms	2x2^6+512	64ms
(0,1,0)	1/8	8x2^6	51.2ms	8x2^6+512	102.4ms
(0,1,1)	1/16	16x2^6	102.40ms	16x2^6+512	153.6ms
(1,0,0)	1/32	32x2^6	204.80ms	32x2^6+512	256ms
(1,0,1)	1/64	64x2^6	409.60ms	64x2^6+512	460.8ms
(1,1,0)	1/128	128x2^6	819.20ms	128x2^6+512	870.4ms
(1,1,1)	1/256	256x2^6	1.638s	256x2^6+512	1.6892s

11.2 Applications of Watchdog Timer Reset

The main application of the Watchdog Timer with time-out reset enabling is for the system monitor. This is important in real-time control applications. In case of some power glitches or electro-magnetic interference, the processor may begin to execute erroneous codes and operate in an unpredictable state. If this is left unchecked the entire system may crash. Using the Watchdog Timer during software development will require the user to select ideal Watchdog reset locations for inserting instructions to reset the Watchdog Timer. By inserting the instruction setting WDCLR, it will allow the code to run

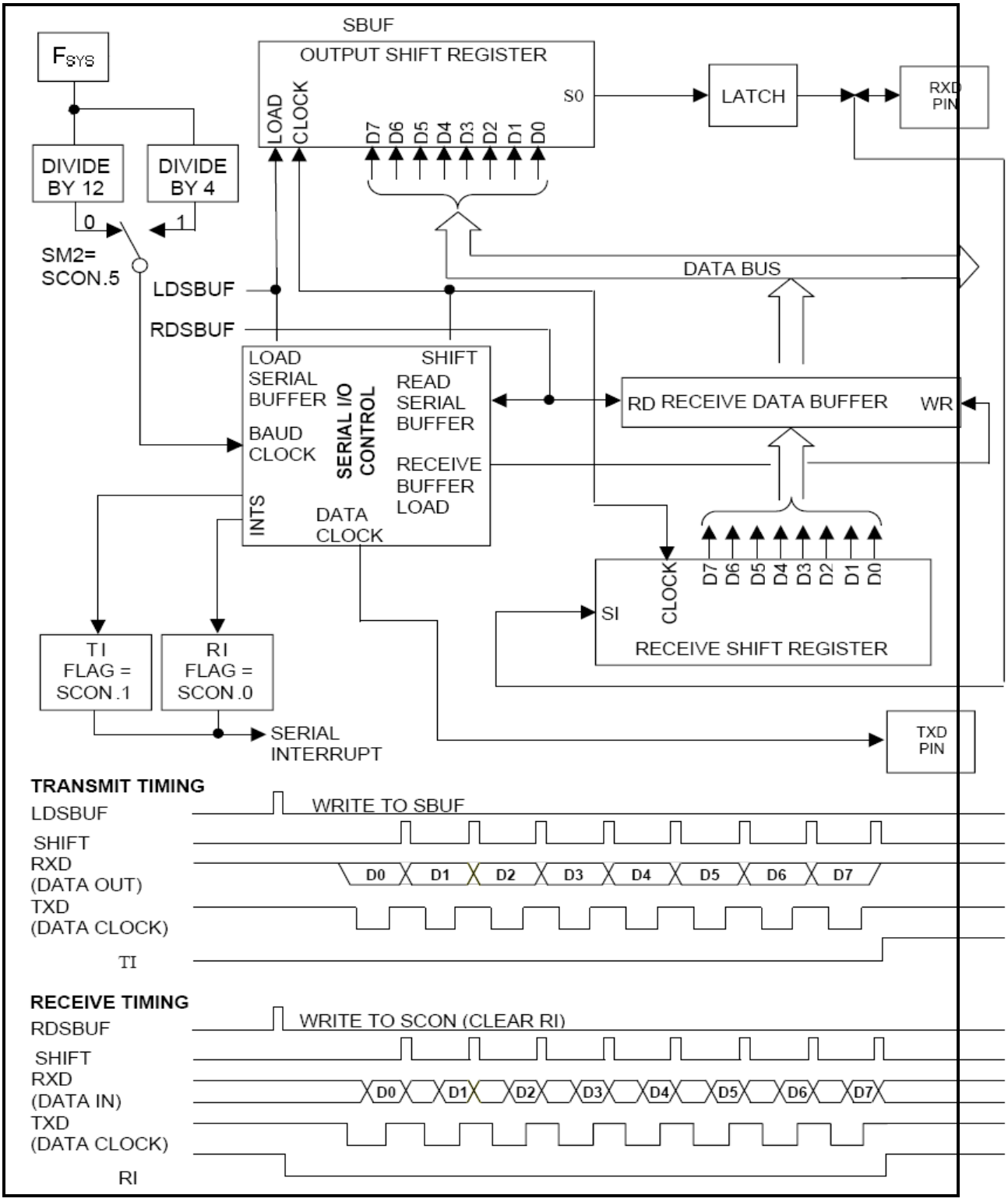


Figure 12-1 Serial Port Mode 0 Function Block

12.6 Framing Error Detection

Framing error detection is provided for asynchronous modes (Mode 1, 2 and 3.) The framing error occurs when a valid stop bit is not detected due to the bus noise or contention. The UART can detect a framing error and notify the software.

The framing error bit, FE, is located in SCON.7. This bit normally serves as SM0. While the framing error detection enable bit SMOD0 (PCON.6) is set 1, it serves as FE flag. Actually SM0 and FE locate in different registers.

The FE bit will be set 1 via hardware while a framing error occurs. It should be cleared via software. Note that SMOD0 should be 1 while reading or writing to FE. If FE is set, any of the following frames received without any error will not clear the FE flag. The clearing has to be done via software.

12.7 Multiprocessor Communication

The communication feature of the N79E715 enables a Master device send a multiple frame serial message to a Slave device in a multi-slave configuration. It does this without interrupting other slave devices that may be on the same serial line. UART mode 2 or 3 mode can use this feature only. After 9 data bits are received. The 9^{th} bit value is written to RB8 (SCON.2). The user can enable this function by setting SM2 (SCON.5) as logic 1 so that when the stop bit is received, the serial interrupt will be generated only if RB8 is 1. When the SM2 bit is 1, serial data frames that are received with the 9^{th} bit as 0 do not generate an interrupt. In this case, the 9^{th} bit simply separates the address from the serial data.

When the Master device wants to transmit a block of data to one of several slaves on a serial line, it first sends out an address byte to identify the target slave. Note that in this case, an address byte differs from a data byte: In an address byte, the 9^{th} bit is 1 and in a data byte, it is 0. The address byte interrupts all slaves so that each slave can examine the received byte and see if it is being addressed. The addressed slave then clears its SM2 bit and prepares to receive incoming data bytes. The SM2 bits of slaves that were not addressed remain set, and they continue operating normally while ignoring the incoming data bytes.

Follow the steps below to configure multiprocessor communications:

1. Set all devices (Masters and Slaves) to UART mode 2 or 3.

2. Write the SM2 bit of all the Slave devices to 1.

3. The Master device's transmission protocol is:

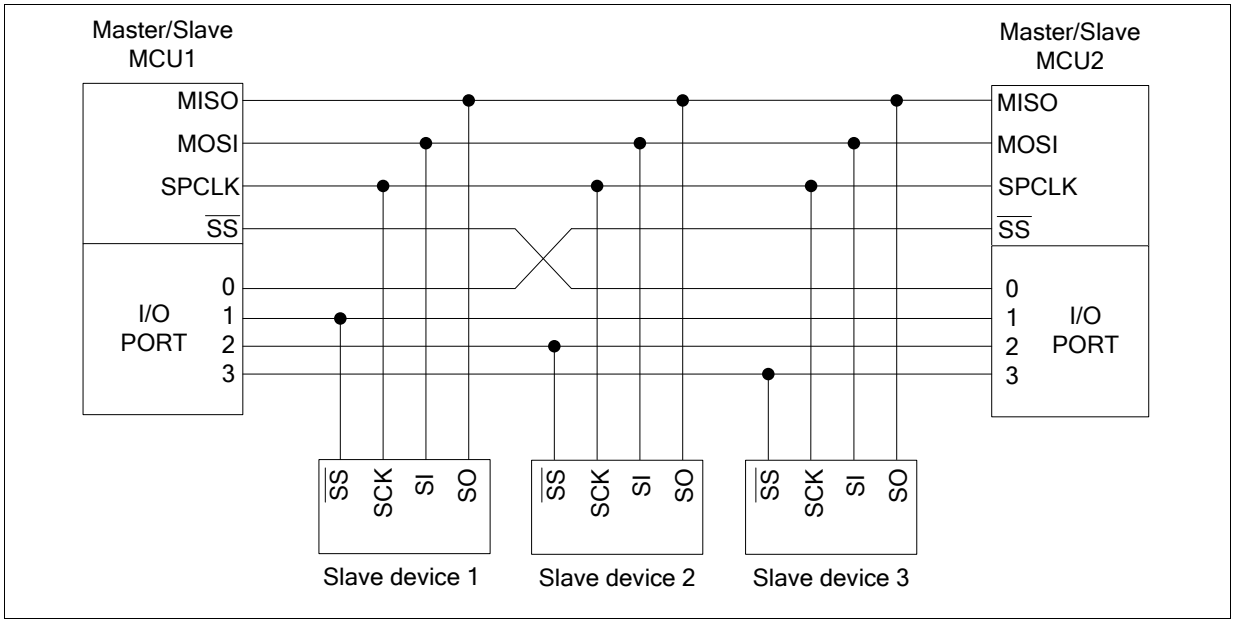


Figure 13-2 SPI Multi-master, Multi-slave Interconnection

Figure 13-2 shows a typical interconnection of SPI devices. The bus generally connects devices together through three signal wires, MOSI to MOSI, MISO to MISO, and SPCLK to SPCLK. The Master devices select the individual Slave devices by using four pins of a parallel port to control the four $\overline{SS}$ pins. MCU1 and MCU2 play either Master or Slave mode. The $\overline{SS}$ should be configured as Master Mode Fault detection to avoid multi-master conflict.

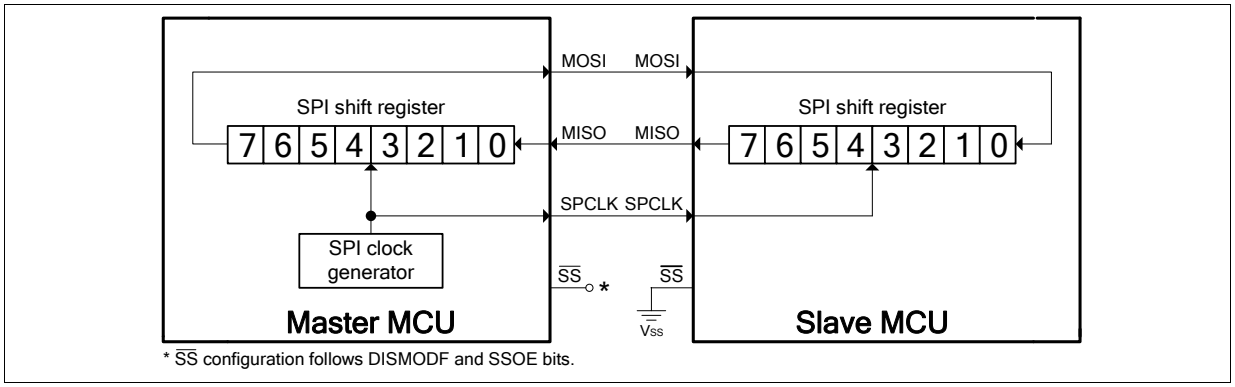


Figure 13-3 SPI Single-master, Single-slave Interconnection

Figure 13-3 shows the simplest SPI system interconnection, single-master and signal-slave. During a transfer, the Master shifts data out to the Slave via MOSI line. While simultaneously, the Master shifts data in from the Slave via MISO line. The two shift registers in the Master MCU and the Slave MCU can be considered as one 16-bit circular shift register. Therefore, while a transfer data pushed from Master into Slave, the data in Slave will also be pulled in Master device respectively. The transfer effectively exchanges the data which was in the SPI shift registers of the two MCUs.

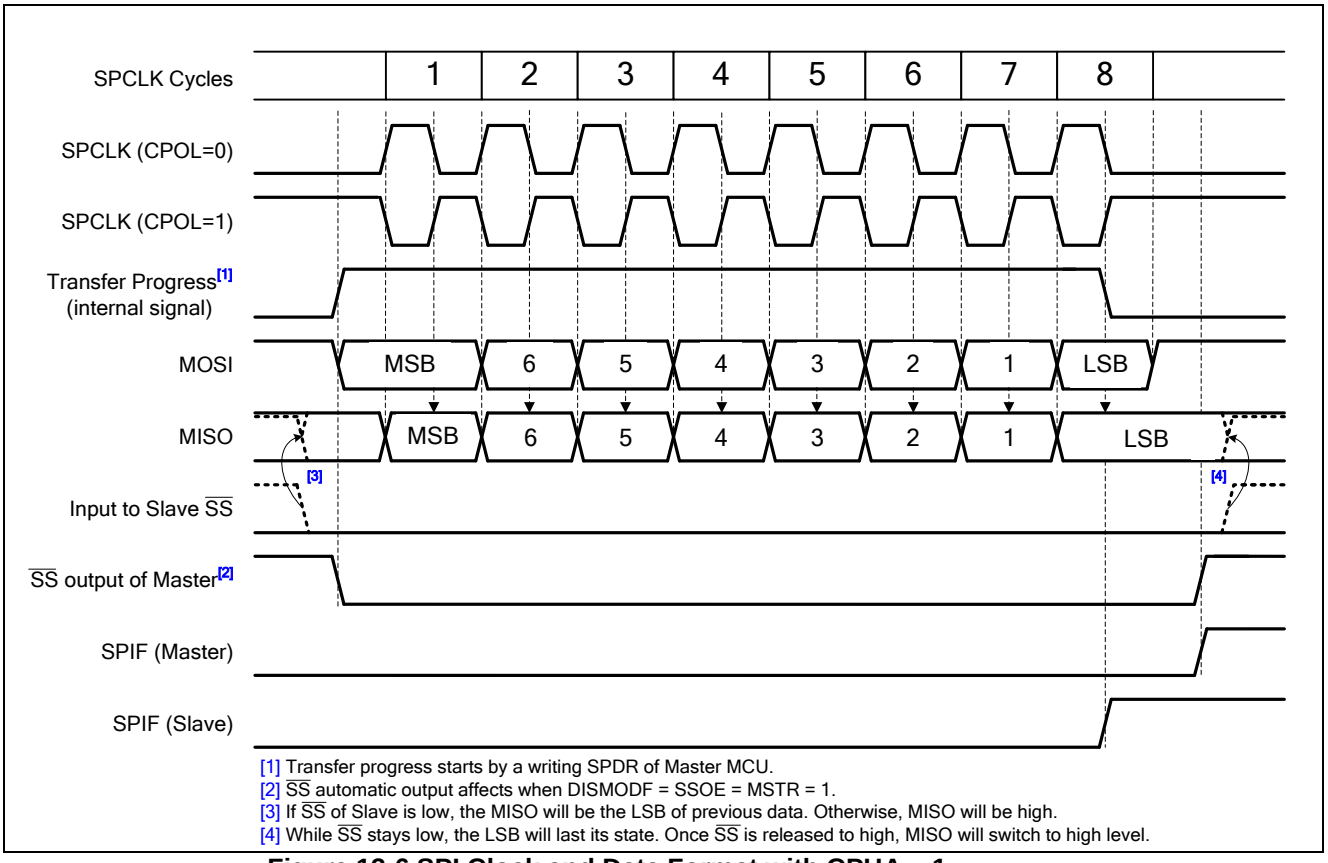


Figure 13-6 SPI Clock and Data Format with CPHA = 1

13.6 Slave Select Pin Configuration

The N79E715 SPI provides a flexible $\overline{SS}$ pin feature for different system requirements. When the SPI operates as a Slave, $\overline{SS}$ pin always rules as Slave select input. When the Master mode is enabled, $\overline{SS}$ has three different functions according to DISMODF (SPSR.3) and SSOE (SPCR.7). By default, DISMODF is 0. It means that the Mode Fault detection activates. $\overline{SS}$ is configured as a input pin to check if the Mode Fault appears. On the contrary, if DISMODF is 1, Mode Fault is inactivated and the SSOE bit takes over to control the function of the $\overline{SS}$ pin. While SSOE is 1, it means the Slave select signal will generate automatically to select a Slave device. The $\overline{SS}$ as output pin of the Master usually connects with the $\overline{SS}$ input pin of the Slave device. The $\overline{SS}$ output automatically goes low for each transmission when selecting external Slave device and goes high during each idle state to de-select the Slave device. While SSOE is 0 and DISMODF is 1, $\overline{SS}$ is no more used by the SPI and reverts to be a general purpose I/O pin.

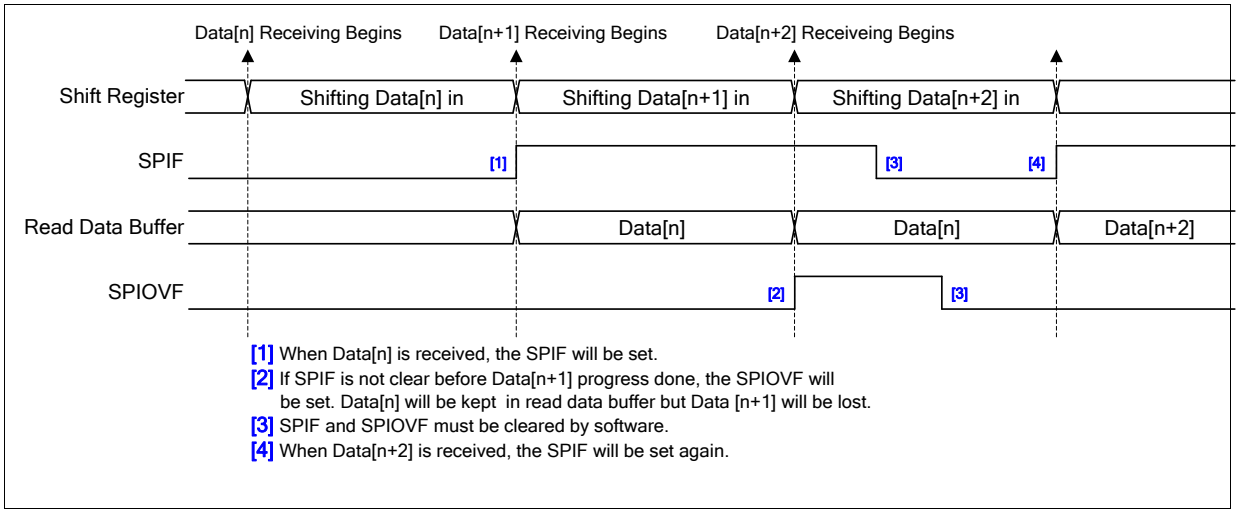


Figure 13-7 SPI Overrun Waveform

13.10 SPI Interrupts

Three SPI status flags, SPIF, MODF, and SPIOVF, can generate an SPI event interrupt requests. All of them locate in SPSR. SPIF will be set after completion of data transfer with external device or a new data have been received and copied to SPDR. MODF becomes set to indicate a low level on $\overline{SS}$ causing the Mode Fault state. SPIOVF denotes a receiving overrun error. If SPI interrupt mask is enabled via setting ESPI (EIE.6) and EA is 1, CPU will executes the SPI interrupt service routine once any of the three flags is set. The user needs to check flags to determine what event caused the interrupt. The three flags are software cleared.

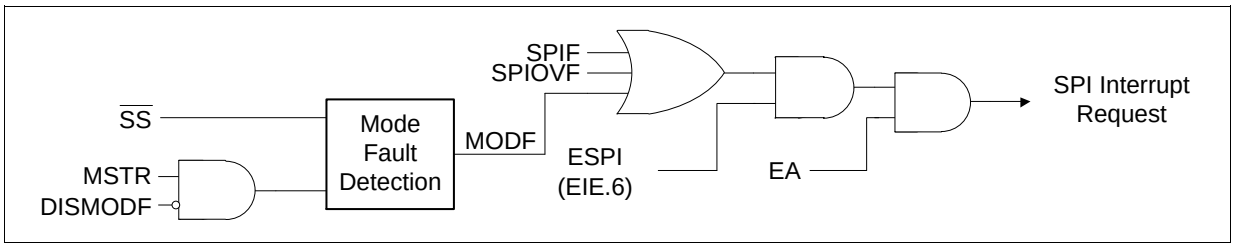


Figure 13-8 SPI Interrupt Request

The successive approximation control logic now sets the next most significant bit (11 0000 0000b or 01 0000 0000b, depending on the previous result), and the V_{DAC} is compared to Vin again. If the input voltage is greater than V_{DAC}, the bit remains set; otherwise it is cleared. This process is repeated until all ten bits have been tested, at which stage the result of the conversion is held in the successive approximation register. The conversion takes four machine-cycles per bit.

The end of the 10-bit conversion is flagged by control bit ADCCON0.4 (ADCI). The upper 8 bits of the result are held in special function register ADCH, and the two remaining bits are held in ADCCON0.7 (ADC.1) and ADCCON0.6 (ADC.0). The user may ignore the two least significant bits in ADCCON0 and use the ADC as an 8-bit converter (8 upper bits in ADCH). In any event, the total actual conversion time is 35 machine-cycles. ADC will be set and the ADCS status flag will be reset 35 cycles after the ADCS is set.

Control bits ADCCON0.0 ~ ADCCON0.2 are used to control an analog multiplexer which selects one of 8 analog channels. An ADC conversion in progress is unaffected by an external or software ADC start. The result of a completed conversion remains unaffected provided ADCI = logic 1; a new ADC conversion already in progress is aborted when entering Idle or Power-down mode. The result of a completed conversion (ADCI = logic 1) remains unaffected when entering Idle mode.

When ADCCON0.5 (ADCEX) is set by external pin to start ADC conversion, after the N79E715 enters Idle mode, P1.4 can start ADC conversion at least ONE machine-cycle.

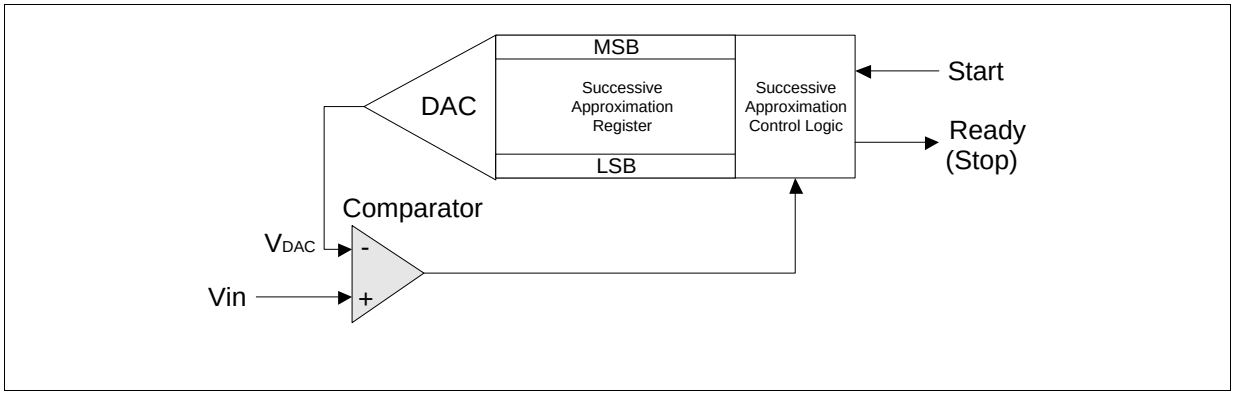


Figure 15-1 Successive Approximation ADC

The ADC circuit has its own supply pins (AV_{DD} and AV_{SS}) and one pins (Vref+) connected to each end of the DAC's resistance-ladder that the AV_{DD} and Vref+ are connected to V_{DD} and AV_{SS} is connected to V_{SS}. The ladder has 1023 equally spaced taps, separated by a resistance of "R". The first tap is located 0.5×R above AV_{SS}, and the last tap is located 0.5×R below Vref+. This gives a total ladder resistance of 1024×R. This structure ensures that the DAC is monotonic and results in a symmetrical quantization error.

For input voltages between AV_{SS} and [(Vref+) + ½ LSB], the 10-bit result of an A/D conversion will be 0000000000B = 000H. For input voltages between [(Vref+) – 3/2 LSB] and Vref+, the result of a conversion will be 1111111111B = 3FFH. Avref+ and AV_{SS} may be between AV_{DD} + 0.2V and AVSS – 0.2 V. Avref+ should be positive with respect to AV_{SS}, and the input voltage (Vin) should be between Avref+ and AV_{SS}.

The result can always be calculated according to the following formula:

$$\text{Result} = 1024 \times \frac{\text{Vin}}{\text{VDD}}$$

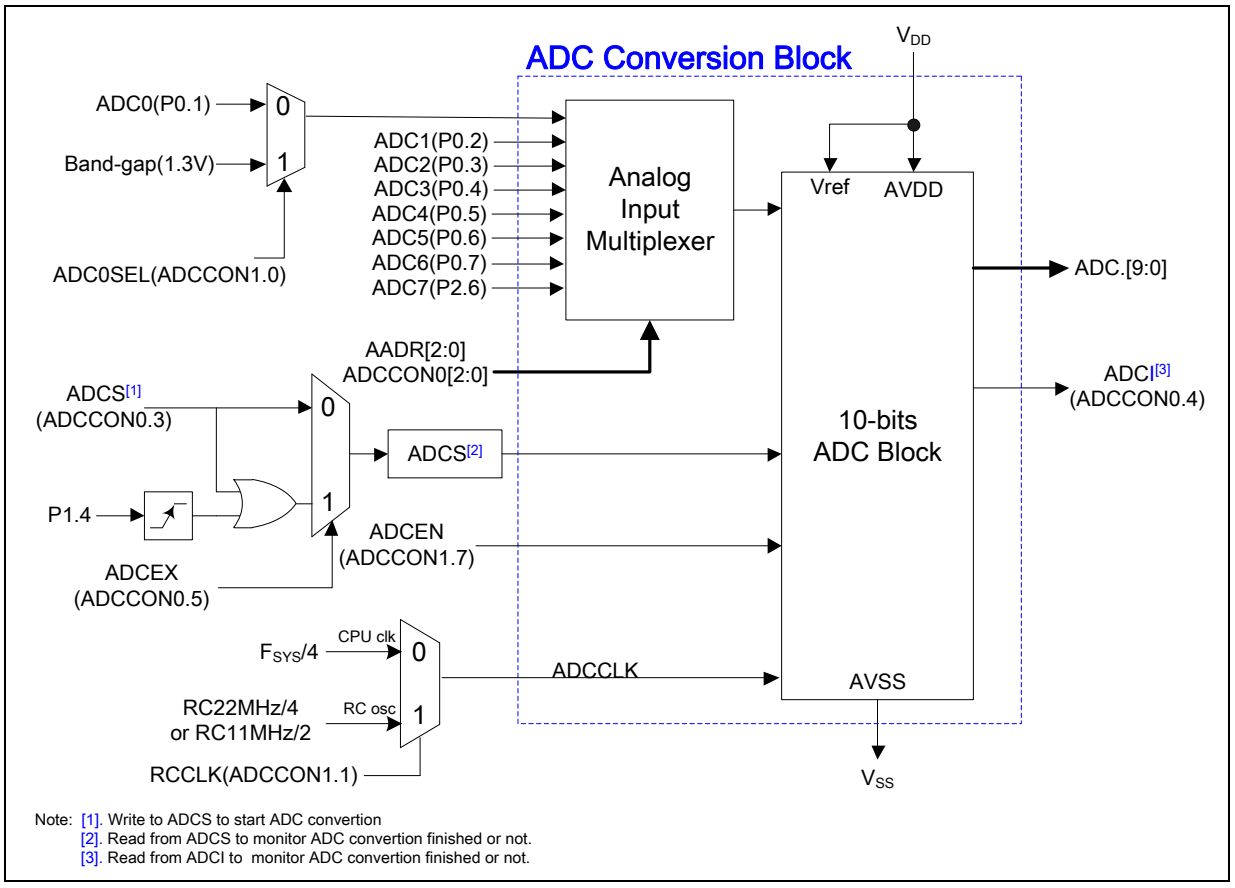


Figure 15-2 ADC Block Diagram

16 Inter-Integrated Circuit (I^2C)

16.1 Features

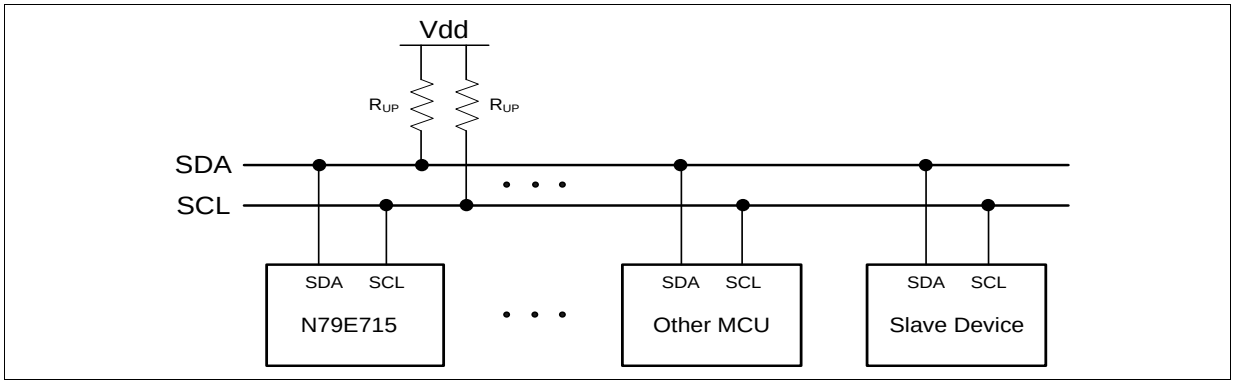
The Inter-Integrated Circuit (I^2C) bus serves as a serial interface between the microcontroller and the I^2C devices such as EEPROM, LCD module, and so on. The I^2C bus used two wires design (a serial data line SDA and a serial clock line SCL) to transfer information between devices.

The I^2C bus uses bidirectional data transfer between masters and slaves. There is no central master and the multi-master system is allowed by arbitration between simultaneously transmitting masters. The serial clock synchronization allows devices with different bit rates to communicate via one serial bus. The I^2C bus supports four transfer modes including master transmitter mode, master receiver mode, slave receiver mode, and slave transmitter mode. The I^2C interface only supports 7-bit addressing mode and General Call can be accepted. The I^2C can meet both standard (up to 100kbps) and fast (up to 400kbps) speeds.

16.2 Functional Description

For the bidirectional transfer operation, the SDA and SCL pins should be connected to open-drain pads. This implements a wired-AND function which is essential to the operation of the interface. A low level on a I^2C bus line is generated when one or more I^2C devices output a “0”. A high level is generated when all I^2C devices output “1”, allowing the pull-up resistors to pull the line high.

In N79E715, the user should set output latches of P1.2 and P1.3. as logic 1 before enabling the I^2C function by setting I2CEN (I2CON.6). The P1.2 and P1.3 are configured as the open-drain I/O once the I^2C function is enabled. The P1M2 and P1M1 will also be re-configured. It is strongly recommended that the Schmitt trigger input buffer be enabled by setting P1S for improved glitch suppression.



```
                //==========
                //Slave Mode
                //==========
                case 0xA0:                          /*A0H,   STOP   or   repeated   START
received while
                                                      still addressed SLAVE mode*/
                      AA = 1;
                      break;
                //======================
                //Slave Transmitter Mode
                //======================
                case 0xA8:                          /*A8H,   own   SLA+R   received,   ACK
returned*/
                      I2DAT = NEXT_SEND_DATA3;
                      AA = 1;                       //when AA is "1", not last data to be
                                                    //transmitted
                      break;
                case 0xB0:                          /*B0H, arbitration lost in SLA+W/R
                                                      own SLA+R received, ACK returned */
                      I2DAT = DUMMY_DATA;
                      AA = 0;                       //when AA is "0", last data to be
                                                    //transmitted
                      STA = 1;                      //retry to transmit START if bus free
                      break;
                case 0xB8:                          /*B8H,   previous   own   SLA+R,   DATA
transmitted,
                                                      ACK received*/
                      I2DAT = NEXT_SEND_DATA4;
                      if (To_TX_Last_Data)               //if   last   DATA   will   be
transmitted
                            AA = 0;
                      else
                            AA = 1;
                      break;
                case 0xC0:                          /*C0H,   previous   own   SLA+R,   DATA
transmitted,
                                                      NACK received, not addressed SLAVE
mode
                                                      entered*/
                      AA = 1;
                      break;
                case 0xC8:                          /*C8H, previous own SLA+R, last DATA
trans-
                                                      mitted, ACK received, not addressed
SLAVE
                                                      mode entered*/
                      AA = 1;
                      break;
          }//end of switch (I2STA)

          SI = 0;                                   //SI should be the last step of I2C
ISR
          while(STO);                               //wait  for  STOP  transmitted  or  bus
error
                                                    //free, STO is cleared by hardware
}//end of I2C_ISR
```

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Address: DFH Reset value: 0000 0000B

Bit	Name	Description
7	BKCH	See the following table (when BKEN is set).
6	BKPS	0 = Brake is asserted if P0.2 is low. 1 = Brake is asserted if P0.2 is high
5	BPEN	See the following table (when BKEN is set).
4	BKEN	0 = Brake is never asserted. 1 = Brake is enabled, and see the following table.
3	PWM3B	0 = PWM3 output is low, when Brake is asserted. 1 = PWM3 output is high, when Brake is asserted.
2	PWM2B	0 = PWM2 output is low, when Brake is asserted. 1 = PWM2 output is high, when Brake is asserted.
1	PWM1B	0 = PWM1 output is low, when Brake is asserted. 1 = PWM1 output is high, when Brake is asserted.
0	PWM0B	0 = PWM0 output is low, when Brake is asserted. 1 = PWM0 output is high, when Brake is asserted.

Brake Condition Table

BPEN	BKCH	BREAK CONDITIONS
0	0	Brake on (software brake and keeping brake)
0	1	On, when PWM is not running (PWMRUN=0), the PWM output condition is follow PWMNB setting. Off, when PWM is running (PWMRUN=1).
1	0	Brake on, when break pin asserted, no PWM output, the bit of PWMRUN will be cleared and BKF flag will be set. The PWM output condition is follow PWMNB setting.
1	1	Not active.

PWMCON2 – PWM Control Register 2

7	6	5	4	3	2	1	0
-	-	-	-	FP1	FP0	-	BKF
-	-	-	-	R/W	R/W	-	R/W

Address: D7H Reset value: 0000 0000B

Bit	Name	Description
7:4	-	**Reserved**

Bit	Name	Description
3:2	FP[1:0]	Select PWM frequency pre-scalar select bits. The clock source of pre-scalar, Fpwm is in phase with F_{SYS} if PWMRUN=1. **FP[1:0] / Fpwm** 00 – F_{SYS} (Default) 01 – F_{SYS} /2 10 – F_{SYS} /4 11 – F_{SYS} /16
1	-	**Reserved**
0	BKF	**External Brake Pin Flag** 0 = PWM is not brake. 1 = PWM is brake by external brake pin. It will be cleared by software.

The Brake function, which is controlled by the contents of the PWMCON1 register, is somewhat unique. In general when Brake is asserted the four PWM outputs are forced to a user selected state, namely the state selected by PWMCON1 bits 0 to 3. As shown in the description of the operation of the PWMCON1 register if PWMCON1.4 is a “1” brake is asserted under the control PWMCON1.7, BKCH, and PWMCON1.5, BPEN. As shown if both are a “0” Brake is asserted. If PWMCON1.7 is a “1” brake is asserted when the run bit, PWMCON0.7, is a “0.” If PWMCON1.6 is a “1” brake is asserted when the Brake Pin, P0.2, has the same polarity as PWMCON1.6. When brake is asserted in response to this pin the RUN bit, PWMCON0.7, is automatically cleared and BKF(PWMCON2.0) flag will be set. The combination of both PWMCON1.7 and PWMCON1.5 being a “1” is not allowed.

Since the Brake Pin being asserted will automatically clear the Run bit of PWMCON0.7and BKF(PWMCON2.0) flag will be set, the user program can poll this bit or enable PWM’s brake interrupt to determine when the Brake Pin causes a brake to occur. The other method for detecting a brake caused by the Brake Pin would be to tie the Brake Pin to one of the external interrupt pins. This latter approach is needed if the Brake signal can be of insufficient length to ensure that it can be captured by a polling routine. When, after being asserted, the condition causing the brake is removed, the PWM outputs go to whatever state that had immediately prior to the brake. This means that to go from brake being asserted to having the PWM run without going through an indeterminate state care should be taken. If the Brake Pin causes brake to be asserted the following prototype code will allow the PWM to go from brake to run smoothly by software polling BKF flag or enable PWM’s interrupt.

Note that if a narrow pulse on the Brake Pin causes brake to be asserted, it may not be possible to go through the above code before the end of the pulse. In this case, in addition to the code shown, an external latch on the Brake Pin may be required to ensure that there is a smooth transition in going from brake to run.

execute an internally generated LCALL instruction which will vector the process to the appropriate interrupt vector address. The conditions for generating the LCALL include:

1. An interrupt of equal or higher priority is not currently being serviced.
2. The current polling cycle is the last machine-cycle of the instruction currently being executed.
3. The current instruction does not involve a write to IE, EIE, IP, IPH, EIP or IPH1 registers and is not a RETI.

If any of these conditions are not met, then the LCALL will not be generated. The polling cycle is repeated every machine-cycle, with the interrupts sampled in the same machine-cycle. If an interrupt flag is active in one cycle but not responded to, and is not active when the above conditions are met, the denied interrupt will not be serviced. This means that active interrupts are not remembered; every polling cycle is new.

The processor responds to a valid interrupt by executing an LCALL instruction to the appropriate service routine. This may or may not clear the flag which caused the interrupt. In case of Timer interrupts, the TF0 or TF1 flags are cleared by hardware whenever the processor vectors to the appropriate timer service routine. In case of external interrupt, INT0 and INT1, the flags are cleared only if they are edge triggered. In case of Serial interrupts, the flags are not cleared by hardware. In the case of Timer 2 interrupt, the flags are not cleared by hardware. The hardware LCALL behaves exactly like the software LCALL instruction. This instruction saves the Program Counter contents onto the Stack, but does not save the Program Status Word PSW. The PC is reloaded with the vector address of that interrupt which caused the LCALL. These address of vector for the different sources are as follows.

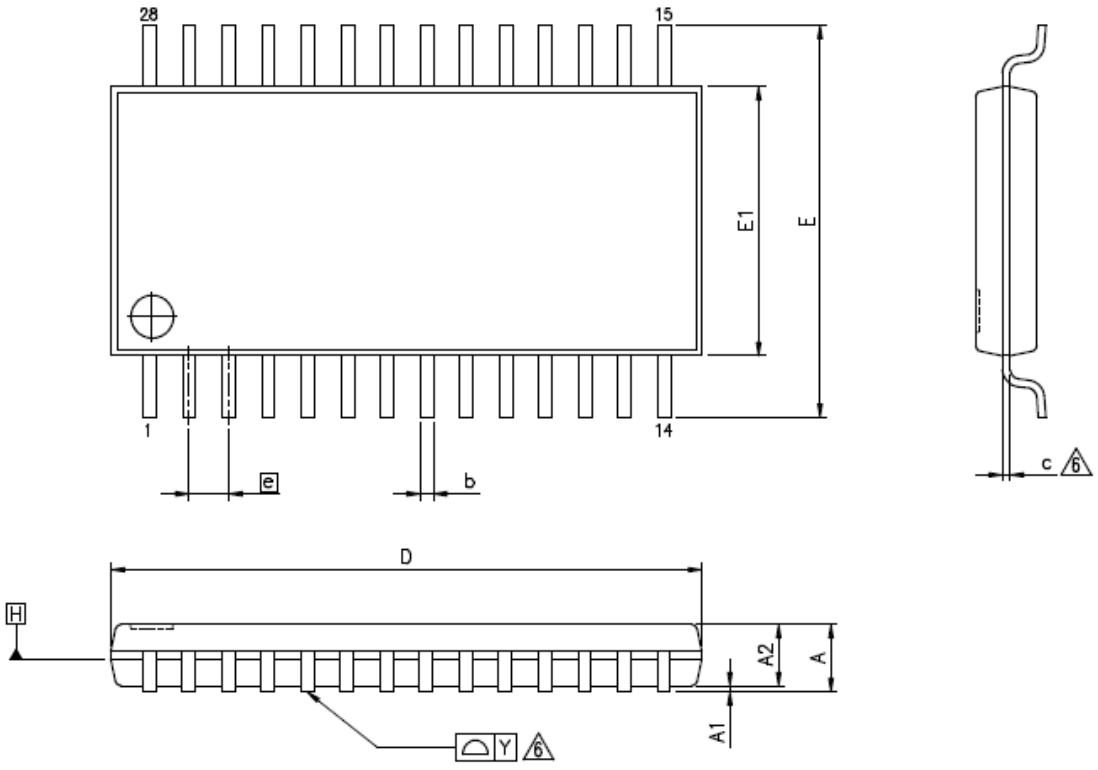
Table 26–1 Instruction Set for N79E715

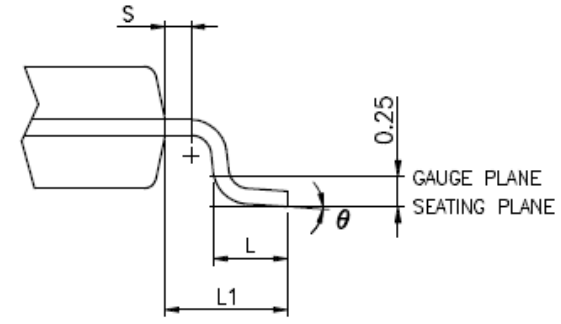
Instruction		OPCODE	Bytes	Clock Cycles	N79E715 vs. Tradition 80C51 Speed Ratio
ORL	C, /bit	A0	2	8	3.0
MOV	C, bit	A2	2	8	1.5
MOV	bit, C	92	2	8	3.0
ACALL	addr11	11, 31, 51, 71, 91, B1, D1, F1[2]	2	12	2.0
LCALL	addr16	12	3	16	1.5
RET		22	1	8	3.0
RETI		32	1	8	3.0
AJMP	addr11	01, 21, 41, 61, 81, A1, C1, E1	2	12	2.0
LJMP	addr16	02	3	16	1.5
JMP	@A+DPTR	73	1	8	3.0
SJMP	rel	80	2	12	2.0
JZ	rel	60	2	12	2.0
JNZ	rel	70	2	12	2.0
JC	rel	40	2	12	2.0
JNC	rel	50	2	12	2.0
JB	bit, rel	20	3	16	1.5
JNB	bit, rel	30	3	16	1.5
JBC	bit, rel	10	3	16	1.5
CJNE	A, direct, rel	B5	3	16	1.5
CJNE	A, #data, rel	B4	3	16	1.5
CJNE	@Ri, #data, rel	B6, B7	3	16	1.5
CJNE	Rn, #data, rel	B8~BF	3	16	1.5
DJNZ	Rn, rel	D8~DF	2	12	2.0
DJNZ	direct, rel	D5	3	16	1.5

[1] The most three significant bits in the 11-bit address [A10:A8] decide the ACALL hex code. The code will be [A10,A9,A8,1,0,0,0,1].

[2] The most three significant bits in the 11-bit address [A10:A8] decide the AJMP hex code. The code will be [A10,A9,A8,0,0,0,0,1].

30.2 28-pin TSSOP - 4.4X9.7 mm

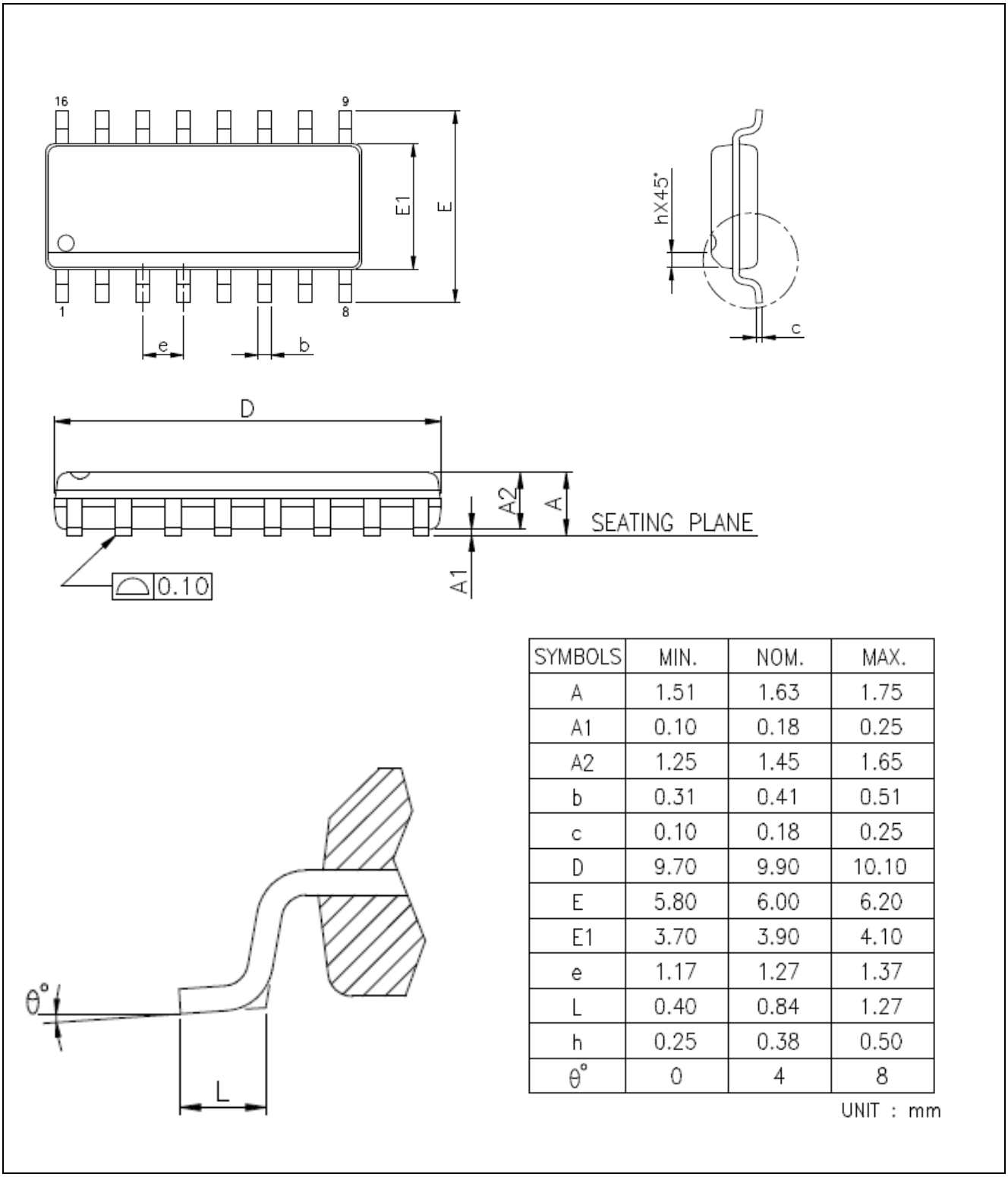




VARIATIONS (ALL DIMENSIONS SHOWN IN MM)

SYMBOLS	MIN.	NOM.	MAX.
A	–	–	1.20
A1	0.00	–	0.15
A2	0.80	1.00	1.05
b	0.19	–	0.30
c	0.09	–	0.20
D	9.60	9.70	9.80
E1	4.30	4.40	4.50
E	6.40 BSC		
e	0.65 BSC		
L1	1.00 REF		
L	0.45	0.60	0.75
S	0.20	–	–
θ	0°	–	8°
Y	0.10		

30.5 16-pin SOP - 150 mil



SYMBOLS	MIN.	NOM.	MAX.
A	1.51	1.63	1.75
A1	0.10	0.18	0.25
A2	1.25	1.45	1.65
b	0.31	0.41	0.51
c	0.10	0.18	0.25
D	9.70	9.90	10.10
E	5.80	6.00	6.20
E1	3.70	3.90	4.10
e	1.17	1.27	1.37
L	0.40	0.84	1.27
h	0.25	0.38	0.50
θ°	0	4	8

UNIT : mm