Nuvoton Technology Corporation - N79E715AS20 Datasheet

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	20-SOIC (0.295", 7.50mm Width)
Supplier Device Package	20-SOP
Purchase URL	https://www.e-xfl.com/product-detail/nuvoton-technology-corporation-america/n79e715as20

Table of Contents

3 Parts Information List

Table 3-1 Lead Free (RoHS) Parts Information List

PART NO.	APROM	LDROM	RAM	DATA FLASH	PACKAGE
N79E715AS28	16KB	2KB	512B	Share APROM	SOP-28 Pin
N79E715AS20	16KB	2KB	512B	Share APROM	SOP-20 Pin
N79E715AS16	16KB	2KB	512B	Share APROM	SOP-16 Pin
N79E715AT28	16KB	2KB	512B	Share APROM	TSSOP-28 Pin
N79E715AT20	16KB	2KB	512B	Share APROM	TSSOP-20 Pin

6 Memory Organization

The N79E715 has embedded Flash EPROM including 16 Kbytes Application Program Flash memory (APROM), fixed 2 Kbytes Load ROM Flash memory (LDROM) and CONFIG-bits. The N79E715 also provides 256 bytes of on-chip direct/indirect RAM and 256 bytes of XRAM accessed by MOVX instruction.

APROM block and Data Flash block comprise the 16 Kbytes embedded Flash. The block size is CONFIG-bits/software configurable.

The N79E715 is built with a CMOS page-erase. The page-erase operation erases all bytes within a page of 128 bytes.

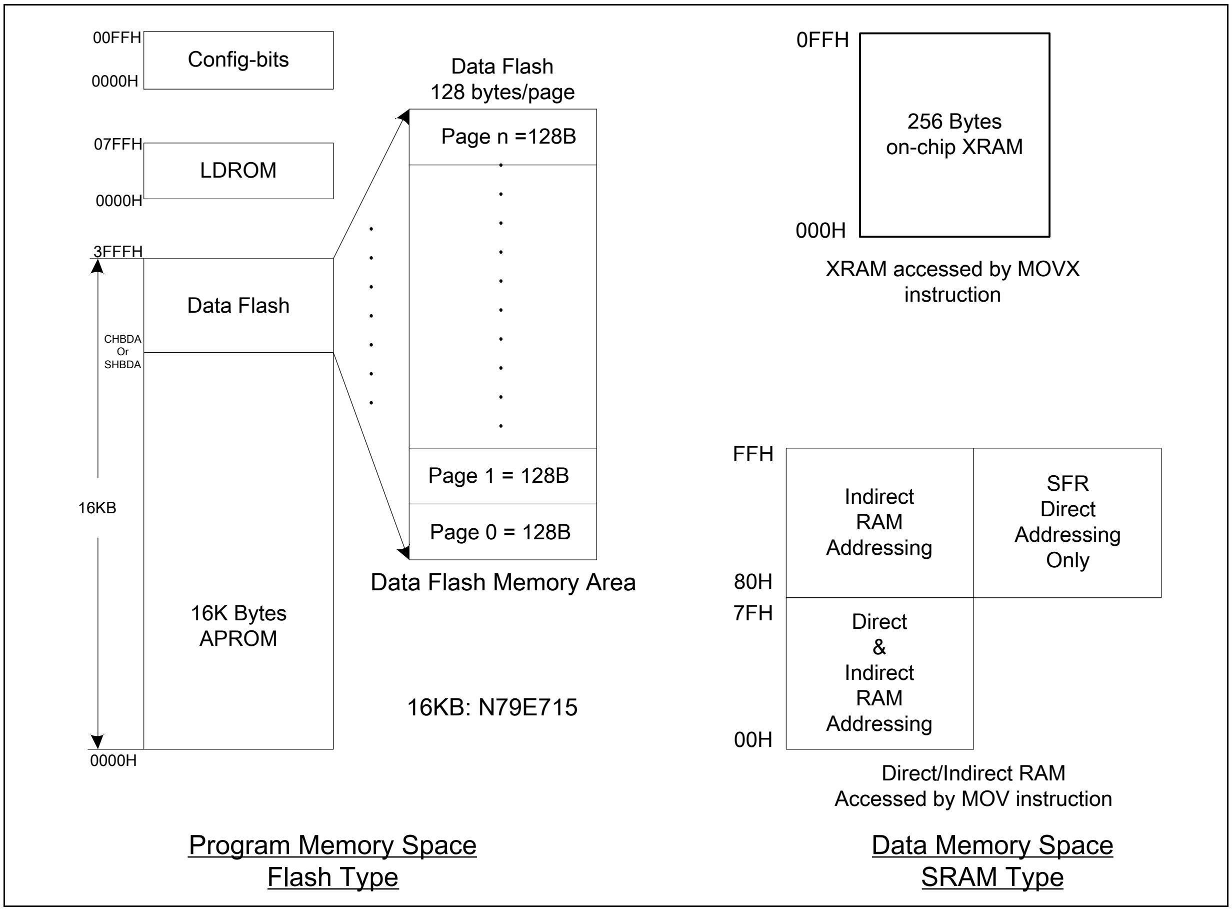

Figure 6-1 N79E715 Memory Map

6.1 APROM Flash Memory

The N79E715 has **16 Kbytes** on-chip Program Memory. All instructions are fetched for execution from this memory area. The MOVC instruction can also read this memory region.

The user application program is located in APROM. When CPU boots from APROM (CHPCON.BS=0), CPU starts executing the program from address 0000H. If the value of program counter (PC) is over the space of APROM, CPU will execute NOP operand and program counter increases one by one until PC reaches 3FFFH then it wraparounds to address 0000H of APROM, the CPU executes the application program again.

6.2 LDROM Flash Memory

The N79E715 has 2 Kbytes LDROM. User may develop the ISP function in LDROM for updating application program or Data Flash. Similarly, APROM can also re-program LDROM and Data Flash. The start address of LDROM is at 0000H corresponding to the physical address of the Flash memory. However, when CPU runs in LDROM, CPU automatically re-vectors the LDROM start address to 0000H, therefore user program regards the LDROM as an independent program memory, meanwhile, with all interrupt vectors that CPU provides.

6.3 CONFIG-bits

There are several bytes of CONFIG-bits located CONFIG-bits block. The CONFIG-bits define the CPU initial setting after power up or reset. Only hardware parallel writer or hardware ICP writer can erase/program CONFIG-bits. ISP program in LDROM can also erase/program CONFIG-bits.

6.4 On-chip Non-volatile Data Flash

The N79E715 additionally has non-volatile Data Flash, which is non-volatile so that it remains its content even after the power is off. Therefore, in general application the user can write or read data which rules as parameters or constants. By the software path, SP mode can erase, written, or read the Data Flash only. Of course, hardware with parallel Programmer/Writer or ICP programmer can also access the Data Flash. The Data Flash size is software adjustable in N79E715 (16 KB) by updating the content of SHBDA. SHBDA[7:0] represents the high byte of 16-bit Data Flash start address and the low byte is hardware set to 00H. The value of SHBDA is loaded from the content of CONFIG1 (CHBDA) after all resets. The application program can dynamically adjust the Data Flash size by resetting SHBDA value. Once the Data Flash size is changed the APROM size is changed accordingly. SHBDA has time access protection while a write to SHBDA is required.

P1 – Port 1 (Bit-addressable)

7	6	5	4	3	2	1	0
P17	P16	-	P14	P13	P12	P11	P10
R/W	R/W	-	R/W	R/W	R/W	R/W	R/W

Address: 90H Reset value: 1111 1111B

Bit	Name	Description
7:0	P1[7:0]	**Port 1** These pins are in quasi-bidirectional mode except P1.2 and P1.3 pins. The P1.2 and P1.3 are dedicating open-drain pins for I^{2}C interface after reset.

P2 – Port 2 (Bit-addressable)

7	6	5	4	3	2	1	0
P27	P26	P25	P24	P23	P22	P21	P20
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Address: A0H Reset value: 1111 1111B

Bit	Name	Description
7:0	P2[7:0]	**Port 2** Port 2 is an 8-bit quasi bidirectional I/O port.

P3 – Port 3 (Bit-addressable)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	P31	P30
-	-	-	-	-	-	R/W	R/W

Address: B0H Reset value: 0000 0011B

Bit	Name	Description
7:2	-	**Reserved**
1	P3.1	X1 or I/O pin by alternative.
0	P3.0	X2 or CLKOUT or I/O pin by alternative.

CKCON – Clock Control

7	6	5	4	3	2	1	0
-	-	-	T1M	T0M	-	-	-
-	-	-	R/W	R/W	-	-	-

Address: 8EH Reset value: 0000 0000B

Bit	Name	Description
7:5	-	**Reserved**
4	T1M	**Timer 1 Clock Selection** 0 = Timer 1 uses a divide by 12 clocks. 1 = Timer 1 uses a divide by 4 clocks.
3	T0M	**Timer 0 Clock Selection** 0 =Timer 0 uses a divide by 12 clocks. 1 = Timer 0 uses a divide by 4 clocks.
2:0	-	**Reserved**

TMOD – Timer 0 and 1 Mode

7	6	5	4	3	2	1	0
GATE	$C/\overline{T}$	M1	M0	GATE	$C/\overline{T}$	M1	M0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Address: 89H Reset value: 0000 0000B

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th></tr>
<tr><td>7</td><td>GATE</td><td><b>Timer 1 Gate Control</b>
0 = Timer 1 will clock when TR1 = 1 regardless of $\overline{INT1}$ logic level.
1 = Timer 1 will clock only when TR1 = 1 and $\overline{INT1}$ is logic 1.</td></tr>
<tr><td>6</td><td>$C/\overline{T}$</td><td><b>Timer 1 Counter/Timer Selection</b>
0 = Timer 1 is incremented by internal peripheral clocks.
1 = Timer 1 is incremented by the falling edge of the external pin T1.</td></tr>
<tr><td>5</td><td>M1</td><td rowspan="2"><b>Timer 1 Mode Selection</b>
<table>
<tr><th>M1</th><th>M0</th><th>Timer 1 Mode</th></tr>
<tr><td>0</td><td>0</td><td>Mode 0: 8-bit Timer/Counter with 5-bit pre-scalar (TL1[4:0])</td></tr>
<tr><td>0</td><td>1</td><td>Mode 1: 16-bit Timer/Counter</td></tr>
<tr><td>1</td><td>0</td><td>Mode 2: 8-bit Timer/Counter with auto-reload from TH1</td></tr>
<tr><td>1</td><td>1</td><td>Mode 3: Timer 1 halted</td></tr>
</table></td></tr>
<tr><td>4</td><td>M0</td></tr>
<tr><td>3</td><td>GATE</td><td><b>Timer 0 Gate Control</b>
0 = Timer 0 will clock when TR0 = 1 regardless of $\overline{INT0}$ logic level.
1 = Timer 0 will clock only when TR0 = 0 and $\overline{INT0}$ is logic 1.</td></tr>
</table>

Figure 10-3 Timer/Counter 0 and 1 in Mode 2

10.1.4 Mode 3 (Two Separate 8-bit Timers)

Mode 3 has different operating methods for the two timers/counters. For timer/counter 1, mode 3 simply freezes the counter. Timer/Counter 0, however, configures TL0 and TH0 as two separate 8 bit count registers in this mode. The logic for this mode is shown in the following figure. TL0 uses the Timer/Counter 0 control bits $C/\overline{T}$, GATE, TR0, $\overline{\text{INT0}}$ and TF0. The TL0 can be used to count clock cycles (clock/12 or clock/4) or 1-to-0 transitions on pin T0 as determined by C/T (TMOD.2). TH0 is forced as a clock cycle counter (clock/12 or clock/4) and takes over the use of TR1 and TF1 from Timer/Counter 1. Mode 3 is used in cases where an extra 8 bit timer is needed. With Timer 0 in Mode 3, Timer 1 can still be used in Modes 0, 1 and 2, but its flexibility is somewhat limited. While its basic functionality is maintained, it no longer has control over its overflow flag TF1 and the enable bit TR1. Timer 1 can still be used as a timer/counter and retains the use of GATE and INT1 pin. In this condition it can be turned on and off by switching it out of and into its own Mode 3. It can also be used as a baud rate generator for the serial port.

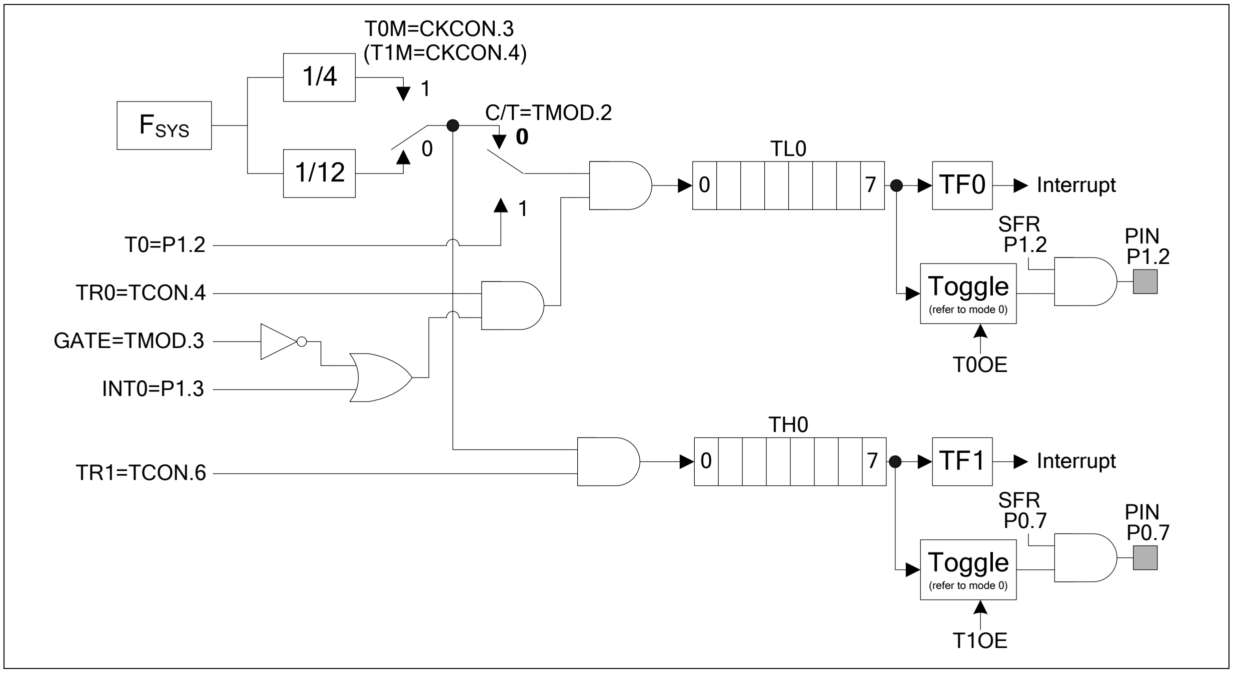

Figure 10-4 Timer/Counter 0 in Mode 3

C1L – Capture 1 Low Byte

7	6	5	4	3	2	1	0
C1L[7:0]							
R/W							

Address: E6H Reset value: 0000 0000B

Bit	Name	Description
7:0	C1L[7:0]	**Input Capture 1 Result Low Byte** The C1L register is the low byte of the 16-bit result captured by input capture 1.

C1H – Capture 1 High Byte

7	6	5	4	3	2	1	0
C1H[7:0]							
R/W							

Address: E7H Reset value: 0000 0000B

Bit	Name	Description
7:0	C1H[7:0]	**Input Capture 1 Result High Byte** The C1H register is the high byte of the 16-bit result captured by input capture 1.

C2L – Capture 2 Low Byte

7	6	5	4	3	2	1	0
C2L[7:0]							
R/W							

Address: EDH Reset value: 0000 0000B

Bit	Name	Description
7:0	C2L[7:0]	**Input Capture 2 Result Low Byte** The C2L register is the low byte of the 16-bit result captured by input capture 2.

C2H – Capture 2 High Byte

7	6	5	4	3	2	1	0
C2H[7:0]							
R/W							

Address: EEH Reset value: 0000 0000B

Bit	Name	Description
7:0	C2H[7:0]	**Input Capture 2 Result High Byte** The C2H register is the high byte of the 16-bit result captured by input capture 2.

10.2.2 Auto-reload Mode

Timer 2 can be configured as auto-reload mode by clearing CP/$\overline{\text{RL2}}$ and setting LDEN bit. In this mode RCOMP2H and RCOMP2L registers stores the reload value. The contents in RCOMP2H and RCOM3L transfer into TH2 and TL2 once the auto-reload event occurs. The event can be the Timer 2

Bit	Name	Description
3	WDTRF	**WDT Reset Flag** When the MCU resets itself, this bit is set by hardware. The bit should be cleared by software. If EWRST = 0, the interrupt flag WDTF won't be set by hardware, and the MCU will reset itself right away. If EWRST = 1, the interrupt flag WDTF will be set by hardware and the MCU will jump into WDT's interrupt service routine if WDT interrupt is enabled, and the MCU won't reset itself until 512 CPU clocks elapse. In other words, in this condition, the user also needs to clear the WDT counter (by writing '1' to WDCLR bit) during this period of 512 CPU clocks, or the MCU will also reset itself when 512 CPU clocks elapse.
2:0	WPS[2:0]	**WDT Pre-scalar Selection** Use these bits to select WDT time-out period. The WDT time-out period is determined by the formula $=\frac{64}{(F_{LIRC} \times Pre\text{-}scalar)}$, where F_{LIRC} is the frequency of the WDT clock source. The following table shows an example of WDT timeout period for different F_{LIRC}.

[1] WDTEN is initialized by reloading the inversed value of CWDTEN (CONFIG3.7) after all resets.
[2] WIDPD and WPS[2:0] are cleared after power-on reset and keep unchanged after any other resets.
[3] WDTRF will be cleared after power-on reset, be set after Watchdog Timer reset, and remains unchanged after any other resets.

WDCON1 – Watchdog Timer Control (TA Protected)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	EWRST
-	-	-	-	-	-	-	R/W

Address: ABH   Reset value: 0000 0000B

Bit	Name	Description
0	EWRST	0 = Disable WDT Reset function. 1 = Enable WDT Reset function.

[1] EWRST is cleared after power-on reset and keeps unchanged after any other resets.

The Watchdog time-out interval is determined by the formula $=\frac{64}{(F_{LIRC} \times Pre\text{-}scalar)}$. Where F_{LIRC} is the frequency of LIRC. The following table shows an example of the Watchdog time-out interval under different F_{LIRC} and pre-scalars.

without any Watchdog Timer reset. However If any erroneous code executes by any power of other interference, the instructions to clear the Watchdog Timer counter will not be executed at the required instants. Thus the Watchdog Timer reset will occur to reset the system start from an erroneously executing condition. The user should remember that WDCON0 requires a timed access writing.

11.3 Applications of Watchdog Timer Interrupt

There is another application of the Watchdog Timer, which is used as a simple timer. The WDTF flag will be set while the Watchdog Timer completes the selected time interval. The software polls the WDTF flag to detect a time-out and the WDCLR allows software to restart the timer. The Watchdog Timer can also be used as a very long timer. Every time the time-out occurs, an interrupt will occur if the individual interrupt EWDI (EIE.4) and global interrupt enable EA is set.

In some application of low power consumption, the CPU usually stays in Idle mode when nothing needs to be served to save power consumption. After a while the CPU will be woken up to check if anything needs to be served at an interval of programmed period implemented by Timer 0, 1 or 2. However, the current consumption of Idle mode still keeps at a “mA” level. To further reducing the current consumption to “µA” level, the CPU should stay in Power-down mode when nothing needs to be served, and has the ability of waking up at a programmable interval. The N79E715 is equipped with this useful function. It provides a very low power LIRC. Along with the low power consumption application, the Watchdog Timer needs to count under Idle and Power-down mode and wake CPU up from Idle or Power-down mode. The demo code to accomplish this feature is shown below.

The demo code of Watchdog Timer wakes CPU up from Power Down.

```
    ORG    0000H
    LJMP   START

    ORG    0053H
    LJMP   WDT_ISR

    ORG    0100H
WDT_ISR:
    CLR    EA
    MOV    TA,#0AAH
    MOV    TA,#55H
    ORL    WDCON0,#01000000B       ;clear Watchdog Timer counter
    INC    ACC
    MOV    P0,ACC
    SETB   EA

    CLR    EA
    MOV    TA,#0AAH
    MOV    TA,#55H
    ANL    WDCON0,#11011111B       ;clear Watchdog Timer interrupt flag
    SETB   EA
    RETI
```

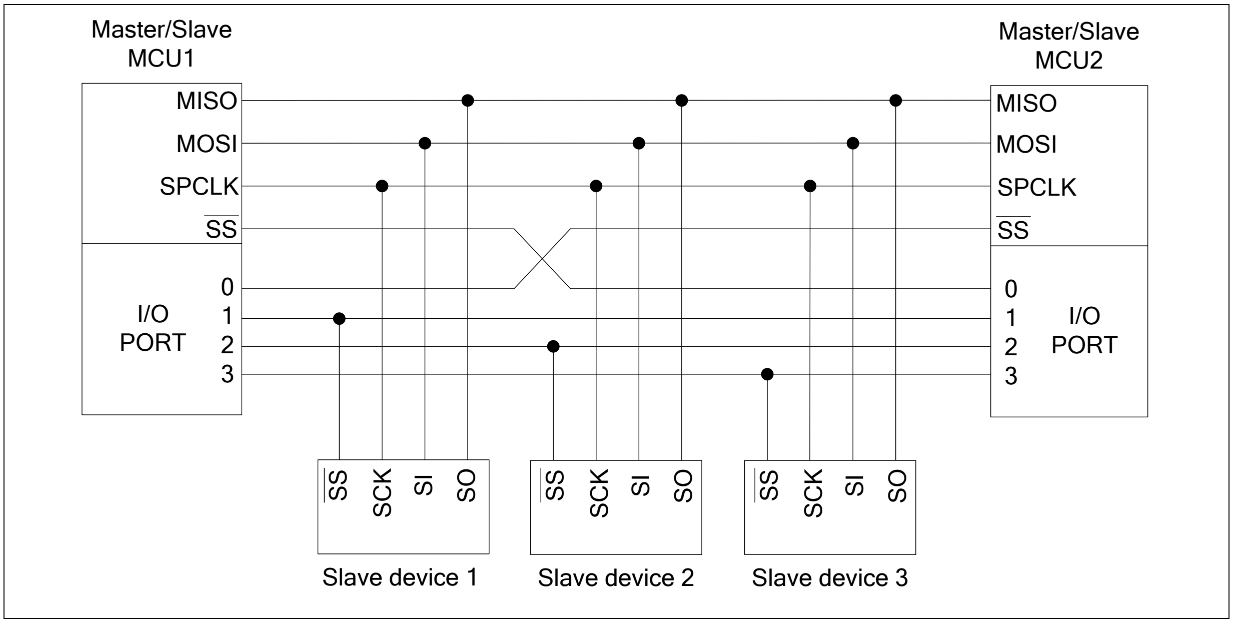

Figure 13-2 SPI Multi-master, Multi-slave Interconnection

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.

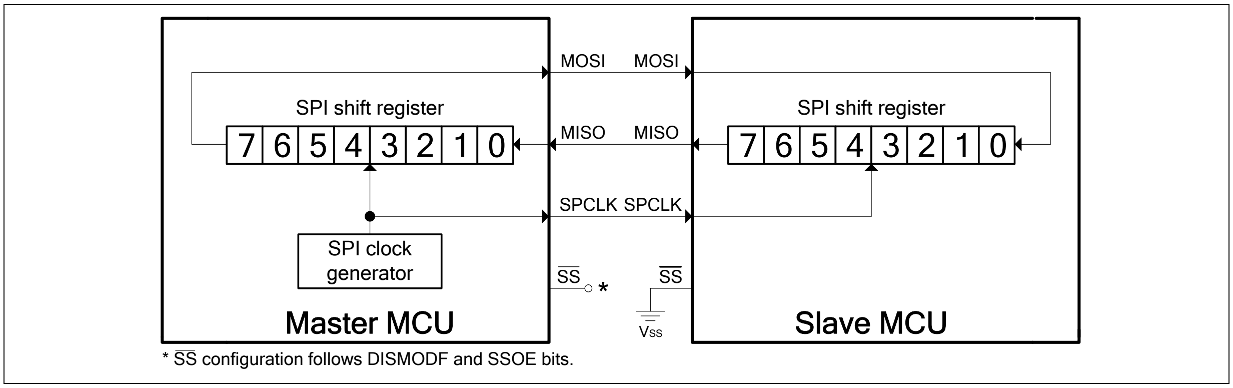

Figure 13-3 SPI Single-master, Single-slave Interconnection

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.

15 Analog-To-Digital Converter (ADC)

The ADC contains a DAC which converts the contents of a successive approximation register to a voltage (V_{DAC}) which is compared to the analog input voltage (Vin). The output of the comparator is fed to the successive approximation control logic which controls the successive approximation register. A conversion is initiated by setting ADCS in the ADCCON0 register. ADCS can be set by software only or by either hardware or software. Note that when the ADC function is disabled, all ADC related SFR bits will be unavailable and will not affect any other CPU functions. The power of ADC block is approached to zero.

The software only start mode is selected when control bit ADCCON0.5 (ADCEX) =0. A conversion is then started by setting control bit ADCCON0.3 (ADCS) The hardware or software start mode is selected when ADCCON0.5 (ADCEX) =1, and a conversion may be started by setting ADCCON0.3 as above or by applying a rising edge to external pin STADC. When a conversion is started by applying a rising edge, a low level should be applied to STADC for at least one machine-cycle followed by a high level for at least one machine-cycle.

The low-to-high transition of STADC is recognized at the end of a machine-cycle, and the conversion commences at the beginning of the next cycle. When a conversion is initiated by software, the conversion starts at the beginning of the machine-cycle which follows the instruction that sets ADCS. ADCS is actually implemented with tpw flip-flops: a command flip-flop which is affected by set operations, and a status flag which is accessed during read operations.

The next two machine-cycles are used to initiate the converter. At the end of the first cycle, the ADCS status flag is set end a value of “1” will be returned if the ADCS flag is read while the conversion is in progress. Sampling of the analog input commences at the end of the second cycle.

During the next eight machine-cycles, the voltage at the previously selected pin of port 0 is sampled, and this input voltage should be stable to obtain a useful sample. In any event, the input voltage slew rate should be less than 10V/ms to prevent an undefined result.

The successive approximation control logic first sets the most significant bit and clears all other bits in the successive approximation register (10 0000 0000b). The output of the DAC (50% full scale) is compared to the input voltage Vin. If the input voltage is greater than V_{DAC}, the bit remains set; otherwise it is cleared.

AUXR1 – AUX Function Resgister-1

7	6	5	4	3	2	1	0
SPI_Sel	UART_Sel	-	-	DisP26	-	0	DPS
R/W	R/W	-	-	R/W	-	R	R/W

Address: A2H Reset value: 0000 0000B

Bit	Name	Description
3	DisP26	0 = Enable P2.6 digital input and output. 1 = Disable P2.6 digital input and output for ADC channel 7 used.

The demo code of ADC channel 0 with clock source = Fsys/4 is as follows:

```
    ORG     0000H
    LJMP    START

    ORG     005BH           ;ADC Interrupt Service Routine
    CLR     ADCI            ;Clear ADC flag
    reti

START:
    ORL     P0DIDS,#02H     ; Disable digital function for P0.1
    ORL     P0M1,#02H       ; ADC0(P0.1) is input-only mode
    ANL     P0M2,#0FDH
    ANL     ADCCON0,#0F8H   ;ADC0(P0.1) as ADC Channel
    ANL     ADCCON1,#0FDH   ;The FSYS/4 clock is used as ADC clock.
    SETB    EADC            ;Enable ADC Interrupt
    SETB    EA
    ORL     ADCCON1,#80H    ;Enable ADC Function

Convert_LOOP:
    SETB    ADCS            ;Trigger ADC
    ORL     PCON,#01H       ;Enter idle mode
    MOV     P0,ADCH         ;Converted Data put in P0 and P1
    MOV     P1,ADCL
    SJMP    Convert_LOOP

    END
```

and the data direction bit “write” (SLA+W). The SI bit should then be cleared to commence SLA+W transaction.

After the SLA+W byte has been transmitted and an acknowledge (ACK) has been returned by the addressed slave device, the SI flag is set again and I2STA is read as 18H. The appropriate action to be taken follows the user defined communication protocol by sending data continuously. After all data is transmitted, the master can send a STOP condition by setting STO (I2CON.4) and then clearing SI to terminate the transmission. A repeated START condition can also be generated without sending STOP condition to immediately initial another transmission.

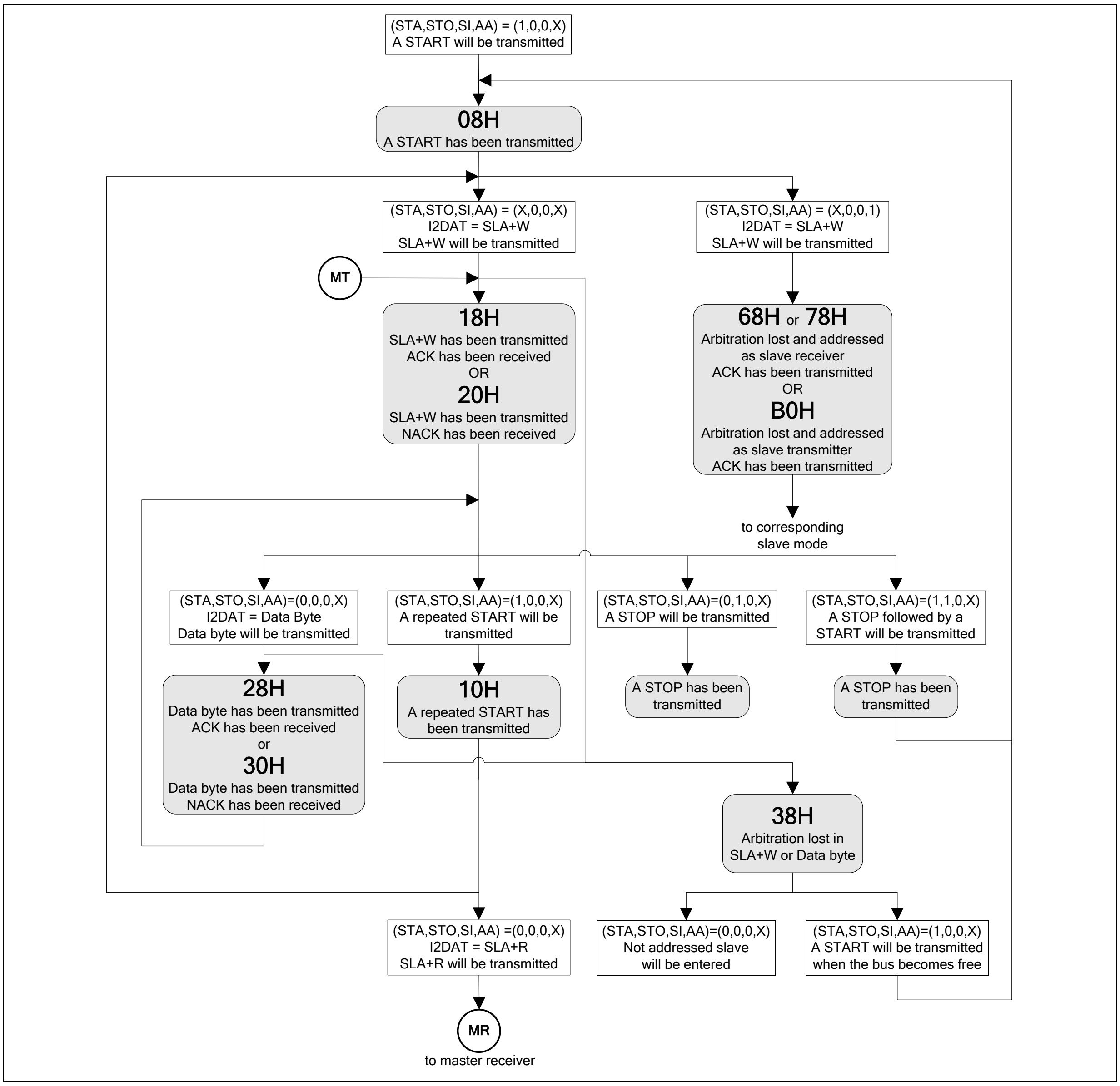
(STA,STO,SI,AA) = (1,0,0,X)
A START will be transmitted
08H
A START has been transmitted
(STA,STO,SI,AA) = (X,0,0,X)
I2DAT = SLA+W
SLA+W will be transmitted
(STA,STO,SI,AA) = (X,0,0,1)
I2DAT = SLA+W
SLA+W will be transmitted
MT
18H
SLA+W has been transmitted
ACK has been received
OR
20H
SLA+W has been transmitted
NACK has been received
68H or 78H
Arbitration lost and addressed as slave receiver
ACK has been transmitted
OR
B0H
Arbitration lost and addressed as slave transmitter
ACK has been transmitted
to corresponding slave mode
(STA,STO,SI,AA)=(0,0,0,X)
I2DAT = Data Byte
Data byte will be transmitted
(STA,STO,SI,AA)=(1,0,0,X)
A repeated START will be transmitted
(STA,STO,SI,AA)=(0,1,0,X)
A STOP will be transmitted
(STA,STO,SI,AA)=(1,1,0,X)
A STOP followed by a START will be transmitted
28H
Data byte has been transmitted
ACK has been received
or
30H
Data byte has been transmitted
NACK has been received
10H
A repeated START has been transmitted
A STOP has been transmitted
A STOP has been transmitted
38H
Arbitration lost in SLA+W or Data byte
(STA,STO,SI,AA) =(0,0,0,X)
I2DAT = SLA+R
SLA+R will be transmitted
(STA,STO,SI,AA)=(0,0,0,X)
Not addressed slave will be entered
(STA,STO,SI,AA)=(1,0,0,X)
A START will be transmitted when the bus becomes free
MR
to master receiver

Figure 16-7 Flow and Status of Master Transmitter Mode

```
             //==========
             //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
```

```
CALL   Enable_ISP
MOV    ISPCN,#11100001b     ;select "CONFIG Program" mode, (A17,A16)=(1,1) for CONFIG
MOV    ISPAH,#00H           ;fill    byte    address,    0000H/0001H/0002H/0003H    for
CONFIG0/1/2/3,
                            ;respectively
MOV    ISPAL,#??H
MOV    ISPFD,#??H           ;fill data to be programmed
CALL   Trigger_ISP

CALL   Disable_ISP
```

CONFIG Read (target address in CONFIG area)

```
CALL   Enable_ISP
MOV    ISPCN,#11000000b     ;select "CONFIG Read" mode, (A17,A16)=(1,1) for CONFIG
MOV    ISPAH,#00H           ;   fill   byte   address,   0000H/0001H/0002H/0003H   for
CONFIG0/1/2/3,
                            ;respectively
MOV    ISPAL,#??H
CALL   Trigger_ISP
MOV    A,ISPFD              ;now,  ISPFD  contains  the  CONFIG  data,  move  to  ACC  for
further
                            ;use
CALL   Disable_ISP
```

23 Power Monitoring

To prevent incorrect execution during power up and power drop, N79E715 provide three power monitor functions, power-on detection and BOD detection.

23.1 Power-on Detection

The power-on detection function is designed for detecting power up after power voltage reaches to a level where system can work. After power-on detected, the POF (PCON.4) will be set 1 to indicate a cold reset, a power-on reset complete. The POF flag can be cleared via software.

23.2 Brown-out Detection

The other power monitoring function, BOD detection circuit is for monitoring the V_{DD} level during execution. There are two programmable BOD trigger levels available for wide voltage applications. The two nominal levels are 2.7V and 3.8V selected via setting CBOV in CONFIG2. When V_{DD} drops to the selected BOD trigger level (V_{BOD}), the BOD detection logic will either reset the CPU or request a BOD interrupt. The user may determine BOD reset or interrupt enable according to different application systems.

The BOD detection will request the interrupt while V_{DD} drops below V_{BOD} while BORST (PMCR.4) is 0. In this case, BOF (PMCR.3) will set as 1. After the user clears this flag whereas V_{DD} remains below V_{BOD}, BOF will not set again. BOF just acknowledge the user a power drop occurs. The BOF will set 1 after V_{DD} goes higher than V_{BOD} to indicate a power resuming. V_{BOD} has a hysteresis of 20~200mV.

CONFIG2

7	6	5	4	3	2	1	0
CBODEN	CBOV	-	CBORST	-	-	-	-
R/W	R/W	-	R/W	-	-	-	-

Factory default value: 1111 1111B

Bit	Name	Description
7	CBODEN	**CONFIG BOD Detection Enable** 1 = Disable BOD detection. 0 = Enable BOD detection. BODEN is initialized by inverted CBODEN (CONFIG2, bit-7) at any resets.

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th></tr>
<tr><td>6</td><td>CBOV</td><td><b>CONFIG BOD Voltage Selection</b>
This bit select one of two BOD voltage level.
<table>
<tr><th>CONFIG-bits
CBOV</th><th>SFR
BOV</th><th>BOD voltage</th></tr>
<tr><td>1</td><td>0</td><td>Enable BOD= 2.7V</td></tr>
<tr><td>0</td><td>1</td><td>Enable BOD= 3.8V</td></tr>
</table></td></tr>
<tr><td>5</td><td>-</td><td><b>Reserved</b></td></tr>
<tr><td>4</td><td>CBORST</td><td><b>CONFIG BOD Reset Enable</b>
This bit decides if a BOD reset is caused after a BOD event.
1 = Enable BOD reset when V_{DD} drops below V_{BOD}.
0 = Disable BOD reset when V_{DD} drops below V_{BOD}.</td></tr>
</table>

PMCR – Power Monitoring Control (TA Protected)

7	6	5	4	3	2	1	0
BODEN	BOV	-	BORST	BOF	-	-	BOS
R/W	R/W	-	R/W	R/W	-	-	R

Address: A3H Reset value: see Table 7–2 N79E715 SFR Description and Reset Values

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th></tr>
<tr><td>7</td><td>BODEN</td><td><b>BOD-detect Function Control</b>
BODEN is initialized by inverted CBODEN (CONFIG2, bit-7) at any resets.
1 = Enable BOD detection.
0 = Disable BOD detection.</td></tr>
<tr><td>6</td><td>BOV</td><td><b>BOD Voltage Select Bits</b>
BOD are initialized at reset with the value of bits CBOV in CONFIG3-bits
BOD Voltage Select bits:
<table>
<tr><th>CONFIG-bits
CBOV</th><th>SFR
BOV</th><th>BOD Voltage</th></tr>
<tr><td>1</td><td>0</td><td>Enable BOD= 2.7V</td></tr>
<tr><td>0</td><td>1</td><td>Enable BOD= 3.8V</td></tr>
</table></td></tr>
<tr><td>5</td><td>-</td><td><b>Reserved</b></td></tr>
<tr><td>4</td><td>BORST</td><td><b>BOD Reset Enable</b>
This bit decides if a BOD reset is caused after a BOD event.
0 = Disable BOD reset when V_{DD} drops below V_{BOD}.
1 = Enable BOD reset when V_{DD} drops below V_{BOD}.</td></tr>
</table>

25 CONFIG Bits (CONFIG)

The N79E715 has several hardware configuration bytes, called CONFIG bits, which are used to configure the hardware options such as the security bits, clock system source, and so on. These hardware options can be re-configured through the Programmer/Writer or ISP modes. N79E715 have four CONFIG bits those are CONFIG0~3. Several functions which are defined by certain CONFIG bits are also available to be re-configured by certain SFR bits. Therefore, there is a need to LOAD such CONFIG bits into respective SFR bits. Such loading will occurs after resets. (Software reset will reload all CONFIG bits except CBS bit in CONFIG0.7) These SFR bits can be continuously controlled via user's software. Other resets will remain the values in these SFR bits unchanged.

Note: CONFIG bits marked as "-" should always keep unprogrammed.

25.1 CONFIG0

7	6	5	4	3	2	1	0
CBS	-	-	-	-	-	LOCK	DFEN
R/W	-	-	-	-	-	R/W	R/W

Factory default value: 1111 1111B

Bit	Name	Description
7	CBS	**CONFIG Boot Selection** This bit defines from which block MCU boots after all resets except software reset. 1 = MCU will boot from APROM after all resets except software reset. 0 = MCU will boot from LDROM after all resets except software reset.
6:2	-	**Reserved**

26 Instruction Sets

The N79E715 executes all the instructions of the standard 8051 family. All instructions are coded within an 8-bit field called an OPCODE. This single byte should be fetched from Program Memory. The OPCODE is decoded by the CPU. It determines what action the microcontroller will take and whether more operation data is needed from memory. If no other data is needed, then only one byte was required. Thus the instruction is called a one byte instruction. In some cases, more data is needed. These will be two or three byte instructions.

Table 26–1 lists all instructions in details. The note of the instruction sets and addressing modes are shown below.

Rn (n = 0~7) Register R0~R7 of the currently selected Register Bank.

Direct 8-bit internal data location’s address. This could be an internal data RAM location (0~127) or a SFR (e.g. I/O port, control register, status register, etc.) (128~255).

@Ri (i = 0, 1) 8-bit internal data RAM location (0~255) addressed indirectly through register R0 or R1.

#data 8-bit constant included in the instruction.

#data16 16-bit constant included in the instruction.

addr16 16-bit destination address. Used by LCALL and LJMP. A branch can be any within the 16 Kbytes Program Memory address space.

addr11 11-bit destination address. Used by ACALL and AJMP. The branch will be within the same 2 Kbytes page of Program Memory as the first byte of the following instruction.

rel Signed (2’s complement) 8-bit offset byte. Used by SJMP and all conditional branches. The range is -128 to +127 bytes relative to first byte of the following instruction.

bit Direct addressed bit in internal data RAM or SFR.

Table 26–1 Instruction Set for N79E715

Instruction	OPCODE	Bytes	Clock Cycles	N79E715 vs. Tradition 80C51 Speed Ratio
NOP	00	1	4	3.0
ADD A, Rn	28~2F	1	4	3.0
ADD A, @Ri	26, 27	1	4	3.0
ADD A, direct	25	2	8	1.5
ADD A, #data	24	2	8	1.5
ADDC A, Rn	38~3F	1	4	3.0