SPI[™] Versus MICROWIRE[™] EEPROM Comparison

Fairchild Application Note 1012



INTRODUCTION

The SPI (Serial Peripheral Interface) provides a simple eight bit serial port useful in communicating with external devices. Prior to the SPI EEPROMs introduction, engineers used the standard MICROWIRE EEPROMs to interface with the SPI port. MICROWIRE's diverse densities and data protection options offered highly versatile solutions. Recently, several manufacturers have developed serial EEPROMs that are now specifically designed to interface with this port. Due to SPI's faster clock speed and interface compatibility, this EEPROM device is increasing in popularity. The following application note will compare these two EEPROMs and where the advantages exist in each family.

To perform this comparison, a 4 kbit SPI and a 4 kbit MICROWIRE EEPROM are interfaced to the SPI port of a HC11. The NM25C04 SPI is a 512 by 8-bit serial EEPROM, while the NM93C66 MICROWIRE is a 256 by 16-bit.

HARDWARE INTERFACE

HC11 microcontroller's can be configured in one of two modes when utilizing the SPI port. The "master" mode sets the microcontroller to orchestrate data transfer to the peripheral device. The "slave" mode allows the peripheral device to command the HC11. In this application, the HC11 is set as the "master" to control the data transfers to and from the serial EEPROMs. Figure 1 and Figure 2 shows the typical SPI and MICROWIRE connection to a HC11 SPI port.

The HC11 SPI port has three lines that control data transfer and one extra general purpose I/O line to control the peripheral device's chip select. The MOSI (Master Out Slave In) line is used as data out, MISO (Master In Slave Out) line is the data input, and the SCK generates the serial clock. The general purpose PD5 line controls the EEPROMs chip select.

Both the SPI and the MICROWIRE devices are configurable as a four wire interface. Thus, neither of the EEPROMs offer a connection advantage.

DATA TRANSFER

The new SPI EEPROMs are specifically designed to interface with the SPI port. Furthermore, it operates at 2.1 MHz versus MICROWIRE's 1 MHz clock. The SPI communication protocol is broken down into byte size sequences containing instruction, address and data transfer information.

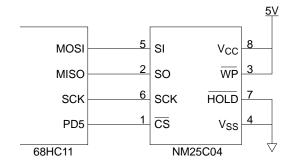


FIGURE 1. NM25C04 to HC11 Connections

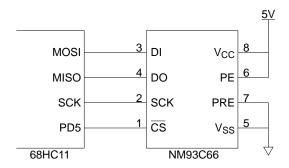


FIGURE 2. NM93C66 to HC11 Connections

Address transfers are dependent on the memory density. Standard MICROWIRE is organized in 16-bit words. To provide the SPI byte size data transfer protocol, three to five bits of additional instructions are required.

Furthermore, a design consideration is required when interfacing a MICROWIRE device to the SPI port. Data is valid relative to the clock signal. Data is written to the EEPROM on the rising edge of the clock. When reading from a MICROWIRE device data is valid on the falling edge.

Development of SPI compatibility is easily accomplished once the design issues are understood. Non-byte wide instruction and address issues can be dealt with by shifting in leading 0's before the required start bit. The leading 0's will not be recognized and can be used to fill out an instruction to byte length. The MICROWIRE word wide architecture can be made to appear byte accessible with clever software. The clock polarity issues can be handled by changing the configuration of the SPI port when receiving or transmitting data.

SPI™ is a trademark of Motorola.

Both devices were driven with a 1 MHz clock from the SPI port. Communications between the HC11 and the serial EEPROMs took roughly the same amount of time.

SOFTWARE COMPARISON

Two software programs are included that demonstrate the interfacing difference between the HC11 microcontroller to a NM93C66 and a NM25C04. The MICROWIRE data transferring differences, as described in the previous section, only required an additional 38 bytes of program space.

The standard MICROWIRE family is organized in a 16-bit (word wide) manner. SPI is an 8-bit (byte wide) structure. Therefore, the additional software overhead that is required for converting a word organization into a byte organization. However, there are devices available at Fairchild that offers the designer the option of either

a x8 or x16 mode. The data size selection is determined by the ORG pin found on the NM93CxxA family. By setting the ORG pin low, the SPI and MICROWIRE devices are comparable in the amount of required programming space.

CONCLUSION

This application note has shown that both the SPI and MICROWIRE families of devices can be effectively interfaced to the HC11. MICROWIRE performance compares favorably with the newer SPI EEPROMs. A few extra bytes of software is the most significant disadvantage when using word wide MICROWIRE mode. In applications where the extra software storage space is available this becomes a non-issue.

When considering availability and price per bit, both families of EEPROMs are quite competitive.

```
**************************
* This code was developed to demonstrate how the NM25C04 serial EEPROM *
* can be interfaced to the MC68HC11 microcontroller. Basic read and
* write operations have been developed. The software demonstrates the
* following commands:
* READ
       :Read a byte
* WREN : Enable write operations
* WRDI : Disable write operations
* WRITE: Program a byte
* The SPI port in Port D is used to interface the NM25C04 to the
* 68HC11. The SPI port provides the clock (SCK), data out (MOSI) and
* the data in (MISO) lines. The 25C04 CS line is driven by a general
* purpose Port D I/O line.
* The mainline was used to test the functionality of the subroutines.
* The subroutines can be copied directly into a customer's program and
* be expected to operate as described. The final mainline only
* performs a write enable, write, write disable and finally a read.
* ADDRESS LOCATION EQUATES
******************
        DDRD
                                $09
                                         port D direction register = $1009
        PORTD
                EQU
                                $08
                                         port D data register = $1008
                                         SPI control register
        SPCR
                                $28
                  EOU
        SPSR
                  EOU
                                $29
                                         SPI status register
                  EOU
                                $2A
                                         SPI data register
* BIT POSITION EQUATES
        CSBITE
                  EOU
                                $20
                                        CS position in port D = bit 5
```

	HIADD	EQU	\$0180	high order page pointer
	LOADD	EQU	\$0181	low order page pointer
	DATVAL	EQU	\$0182	data transfer register
****	*****	* * * * * * * * * * * * *	******	***********
* RESET	VECTOR			
*				***********
	ORG	\$FFFE		reset vector to \$E000
	FDB	\$E000		·
*****	*****	*****	*****	************
* PROGR	AM STARTING	LOCATION		
*****	*****	******	******	************
	ODG	4 7000		77000
BEGIN:	ORG LDS	\$E000		program execution begins at \$E000
BEGIN.	LDX	#\$01FF #\$1000		initialize stack pointer initialize ≤address index register≤
	LDAA	#\$1000 #\$EF		initialize I/O ports (CS = 1, SCK = 0)
	STAA	PORTD, X		initialize 1/0 poits (CS = 1, BCR = 0)
	LDAA	#\$3F		
	STAA	DDRD, X		SPI bits set to outputs
	LDAA	#\$54		DIT DIED DEC CO Output
	STAA	SPCR, X		initialize SPI port CPOL = 0, CPHA = 1
	LDAA	SPSR, X		reset SPIF bit
*****	*****		*****	**********
* MAINL	INE			
*****	*****	*****	******	************
	JSR	WREN		enable write operations
	LDAA	#\$01		
	STAA	HIADD		
	LDAA	#\$23		
	STAA	LOADD		address = 123H
	LDAA	#\$96		
	STAA	DATVAL		data = 96H
	JSR	WRITE		write data 96H into address 0123H
	JSR	WRDI		disable writes
	LDAA	#\$01		
	STAA	HIADD		
	LDAA	#\$23		11 100**
	STAA	LOADD		address = 123H
	JSR	READ		read address 123H
LOOP:	BRA ******	LOOP	********	<pre>wait until reset loop ***********************************</pre>
		AL ROUTINES		
			*****	************

³

execu	ted succes	sfully.			*
*****	******	*****	*****	**********	*****
RITE:	BCLR	PORTD, X#	CSBIT	enable device	
	LDAA	HIADD			
	ASLA			move high order address to	
	ASLA			proper bit location	
	ASLA				
	ORAA	#\$02		OR in WRITE instruction	
	JSR	SENDB		send WRITE instruction	
	LDAA	LOADD			
	JSR	SENDB		send in low order address	
	LDAA	DATVAL			
	JSR	SENDB		send in data value	
	BSET	PORTD, X	#CSBIT	disable device	
	JSR	BUSY		wait until write has completed	
	RTS				
*****	******	* * * * * * * * * * * * * *	* * * * * * * * * * * * * *	*************	******
READ	performs a	byte read ope	ration from th	he NM25C04. The routine	*
expec	ts the add	ress to read t	o be specified	d in the HIADD and LOADD	*
' varia	bles. The	data in the sp	ecified addres	ss is returned in the	*
DATVA	L variable	•			*
*****	*****	******	******	************	*******
READ:	BCLR	PORTD, X	#CSBIT	enable device	
READ:	BCLR LDAA	PORTD, X HIADD	#CSBIT	enable device	
READ:			#CSBIT	move high order address to	
READ:	LDAA		#CSBIT		
READ:	LDAA ASLA		#CSBIT	move high order address to proper bit location	
READ:	LDAA ASLA ASLA		#CSBIT	move high order address to proper bit location OR in READ instruction	
READ:	LDAA ASLA ASLA ASLA	HIADD	#CSBIT	move high order address to proper bit location	
READ:	LDAA ASLA ASLA ORAA	HIADD #\$03	#CSBIT	move high order address to proper bit location OR in READ instruction	
READ:	LDAA ASLA ASLA ASLA ORAA JSR	HIADD #\$03 SENDB	#CSBIT	move high order address to proper bit location OR in READ instruction	
	LDAA ASLA ASLA ORAA JSR LDAA	#\$03 SENDB LOADD	#CSBIT	move high order address to proper bit location OR in READ instruction send READ instruction	
	LDAA ASLA ASLA ORAA JSR LDAA JSR	#\$03 SENDB LOADD	#CSBIT	move high order address to proper bit location OR in READ instruction send READ instruction send in low order address	
	LDAA ASLA ASLA ORAA JSR LDAA JSR SENDB	#\$03 SENDB LOADD SENDB	#CSBIT	move high order address to proper bit location OR in READ instruction send READ instruction send in low order address	
rSR	LDAA ASLA ASLA ORAA JSR LDAA JSR SENDB STAA BSET	#\$03 SENDB LOADD SENDB DATVAL PORTD, X	#CSBIT	move high order address to proper bit location OR in READ instruction send READ instruction send in low order address read byte from NM25C04 disable device	
JSR	LDAA ASLA ASLA ORAA JSR LDAA JSR SENDB STAA BSET	#\$03 SENDB LOADD SENDB DATVAL PORTD, X	#CSBIT	move high order address to proper bit location OR in READ instruction send READ instruction send in low order address read byte from NM25C04	****
JSR ****** * WREN	LDAA ASLA ASLA ORAA JSR LDAA JSR SENDB STAA BSET	#\$03 SENDB LOADD SENDB DATVAL PORTD, X	#CSBIT ************* erform a write	move high order address to proper bit location OR in READ instruction send READ instruction send in low order address read byte from NM25C04 disable device	*
* WREN	LDAA ASLA ASLA ORAA JSR LDAA JSR SENDB STAA BSET	#\$03 SENDB LOADD SENDB DATVAL PORTD, X	#CSBIT ************* erform a write	move high order address to proper bit location OR in READ instruction send READ instruction send in low order address read byte from NM25C04 disable device	*
JSR ****** * WREN * funct * preve	LDAA ASLA ASLA ORAA JSR LDAA JSR SENDB STAA BSET ******** enables th ion along nt against	#\$03 SENDB LOADD SENDB DATVAL PORTD, X ************* e NM25C04 to p with the WRDI inadvertant d	#CSBIT ******** erform a write (write disable ata changes.	move high order address to proper bit location OR in READ instruction send READ instruction send in low order address read byte from NM25CO4 disable device ***********************************	* *
JSR ****** * WREN * funct * preve	LDAA ASLA ASLA ORAA JSR LDAA JSR SENDB STAA BSET ******** enables th ion along nt against	#\$03 SENDB LOADD SENDB DATVAL PORTD, X ************* e NM25C04 to p with the WRDI inadvertant d	#CSBIT ******** erform a write (write disable ata changes.	move high order address to proper bit location OR in READ instruction send READ instruction send in low order address read byte from NM25C04 disable device	* *
JSR ****** * WREN * funct * preve ******	LDAA ASLA ASLA ORAA JSR LDAA JSR SENDB STAA BSET *********** enables th ion along nt against *********	#\$03 SENDB LOADD SENDB DATVAL PORTD, X *************** e NM25C04 to p with the WRDI inadvertant d ***********	#CSBIT ************ erform a write (write disable ata changes. *******	move high order address to proper bit location OR in READ instruction send READ instruction send in low order address read byte from NM25C04 disable device ***********************************	* *
JSR ****** * WREN * funct * preve ******	LDAA ASLA ASLA ORAA JSR LDAA JSR SENDB STAA BSET *********** enables th ion along nt against ********** BCLR	#\$03 SENDB LOADD SENDB DATVAL PORTD, X ************** e NM25C04 to p with the WRDI inadvertant d ************************************	#CSBIT ******** erform a write (write disable ata changes.	move high order address to proper bit location OR in READ instruction send READ instruction send in low order address read byte from NM25CO4 disable device ***********************************	* *
JSR ****** * WREN * funct * preve	LDAA ASLA ASLA ORAA JSR LDAA JSR SENDB STAA BSET ********* enables th ion along int against ********* BCLR LDAA	#\$03 SENDB LOADD SENDB DATVAL PORTD, X *************** e NM25C04 to p with the WRDI inadvertant d ***********	#CSBIT ************ erform a write (write disable ata changes. ********	move high order address to proper bit location OR in READ instruction send READ instruction send in low order address read byte from NM25C04 disable device ***********************************	* *
JSR ****** * WREN * funct * preve ******	LDAA ASLA ASLA ORAA JSR LDAA JSR SENDB STAA BSET *********** enables th ion along nt against ********** BCLR	#\$03 SENDB LOADD SENDB DATVAL PORTD, X ************** e NM25C04 to p with the WRDI inadvertant d ************************************	#CSBIT ************ erform a write (write disable ata changes. ********	move high order address to proper bit location OR in READ instruction send READ instruction send in low order address read byte from NM25C04 disable device ***********************************	* *
JSR ****** * WREN * funct * preve ******	LDAA ASLA ASLA ORAA JSR LDAA JSR SENDB STAA BSET ********* enables th ion along int against ********* BCLR LDAA	#\$03 SENDB LOADD SENDB DATVAL PORTD, X *********** e NM25C04 to p with the WRDI inadvertant d ************* PORTD, X #\$06	#CSBIT ************ erform a write (write disable ata changes. ********	move high order address to proper bit location OR in READ instruction send READ instruction send in low order address read byte from NM25C04 disable device ***********************************	* *
JSR ****** * WREN * funct * preve ******	LDAA ASLA ASLA ORAA JSR LDAA JSR SENDB STAA BSET ********* enables th ion along int against ********* BCLR LDAA JSR	#\$03 SENDB LOADD SENDB DATVAL PORTD, X *********** e NM25C04 to p with the WRDI inadvertant d ************ PORTD, X #\$06 SENDB	#CSBIT ******** erform a write (write disable ata changes. ***********************************	move high order address to proper bit location OR in READ instruction send READ instruction send in low order address read byte from NM25C04 disable device ***********************************	* *
TSR ****** ** WREN * funct * preve *****	LDAA ASLA ASLA ORAA JSR LDAA JSR SENDB STAA BSET ******** enables th ion along nt against ********* BCLR LDAA JSR BSET	#\$03 SENDB LOADD SENDB DATVAL PORTD, X *********** e NM25C04 to p with the WRDI inadvertant d ************ PORTD, X #\$06 SENDB	#CSBIT ******** erform a write (write disable ata changes. ***********************************	move high order address to proper bit location OR in READ instruction send READ instruction send in low order address read byte from NM25C04 disable device ***********************************	* *
****** WREN funct preve *****	LDAA ASLA ASLA ORAA JSR LDAA JSR SENDB STAA BSET ********* enables th ion along nt against ******** BCLR LDAA JSR BSET RTS	#\$03 SENDB LOADD SENDB DATVAL PORTD, X *********** e NM25C04 to p with the WRDI inadvertant d ********** PORTD, X #\$06 SENDB PORTD, X	#CSBIT *********** erform a write (write disable ata changes. ********** #CSBIT #CSBIT	move high order address to proper bit location OR in READ instruction send READ instruction send in low order address read byte from NM25C04 disable device ***********************************	* * * ******

```
WRDI:
       BCLR
                 PORTD, X
                             #CSBIT enable device
                #$04
       LDAA
       JSR
                 SENDB
                                      send EWDS instruction
                             #CSBIT
       BSET
                 PORTD, X
                                       disable device
       RTS
**************************
* BUSY is used to pause until a write operation has completed.
BUSY:
       BCLR
                PORTD, X#
                             CSBIT
       LDAA
                #$05
       JSR
                SENDB
                                       send ≤read status reg≤ instruction
       JSR
                SENDB
                                       read status register
                PORTD, X
       BSET
                            #CSBIT
       ANDA
                 #$01
       BNE
                 BUSY
                                       loop until RDY bit in status is low
       RTS
*************************
* SENDB is used to send a byte to the NM25C04 and also read the data
* that has been returned to the 68HC11.
**************************
SENDB: STAA
                SPDR, X
                                       send byte in A register
                            #$80 PAUSE wait until byte has been sent
PAUSE: BRCLR
                SPSR, X
       LDAA
                 SPDR, X
                                       read byte into a register
       RTS
* This code was developed to demostrate how the NM93C66 serial EEPROM
* can be interfaced to the MC68HC11 microcontroller SPI port. The
* software includes several subroutines that perform various interface
* functions. The internal architecture of the NM93C66 is configured
* with word wide data registers. This applications code demonstrates
* storage of data in a byte wide manner. Odd byte addresses are stored
* in the D8-D15 bits of each word and even address data is stored in
* the D0-D7 bits. The software demonstrates the following commands:
* READ : Read a byte
* WEN
       :Enable write and erase operations
       :Disable write and erase operations
* WDS
* WRITE : Program a byte
* The 68HC11 interfaces to the NM93C66A by using 3 lines from the SPI
* port and a general purpose I/O port bit. The 68HC11 SCK, MOSI and
* MISO pins are used to drive the NM93C66 SK, DI and DO pins
* respectively. Port D bit 5 is used to drive the CS line of the
* The mainline was used to test the functionality of the subroutines.
* The subroutines can be copied directly into a customer's program and
* be expected to operate as described. The final mainline only
```

*****	* * * * * * * * * *	* * * * * * * * * * * * * * *	* * * * * * * * * * * * * * *	*************
* ADDRES	SS LOCATION	N EQUATES		
*****	*****	* * * * * * * * * * * * * * *	* * * * * * * * * * * * * *	**************
	DDRD	EQU	\$09	port D direction register = \$1009
	PORTD	EQU	\$08	port D data register = \$1008
	SPCR	EQU	\$28	SPI control register
	SPSR	EQU	\$29	SPI status register
	SPDR	EQU	\$2A	SPI data register
*****	* * * * * * * * * *	******	******	**********
* BIT P	OSITION EQ	JATES		
^ *****	* * * * * * * * * *	******	*****	*************
	CODIT	FOI	400	CO contribution in contribution in the Contrib
	CSBIT	EQU	\$20	CS position in port D = bit 5
			*****	*************
	BLE ADDRES:	~	*****	**********
	HIADD	EQU	\$0180	high order page pointer
	LOADD	EQU	\$0181	low order page pointer
	DATVAL	EQU	\$0182	data transfer register
	TEMP	EQU	\$0183	temporary ≤scratch≤ register
*****	* * * * * * * * * *	*****	*****	***********
* RESET	VECTOR			
*			***	******************
****	****	*****	*****	**************
	ORG	\$FFFE		reset vector to \$E000
	FDB	\$E000		
*****	* * * * * * * * * *	******	*****	**************
	AM STARTING			
*****	*****	******	*****	************
	ORG	\$E000		program execution begins at \$E000
BEGIN:	LDS	#\$01FF		initialize stack pointer
	LDX	#\$1000		initialize ≤address index register≤
	LDAA	#\$CF		initialize I/O ports (CS = 0, SCK = 0)
	STAA	PORTD, X		
	LDAA	#\$3F		
	STAA	DDRD, X		SPI bits set to outputs
	LDAA	#\$50		
	STAA	SPCR, X	initialize	SPI port
	LDAA	SPSR, X	reset SPIF	bit

	TOD	T.TD.TD.T	
	JSR	WREN	enable write operation
	LDAA	#\$01	
	STAA	HIADD	
	LDAA	#\$23	
	STAA	LOADD	
	LDAA	#\$96	
	STAA	DATVAL	
	JSR	WRITE	write data 96H into address 0123H
	JSR	WDS	disable writes
	LDAA	#\$01	
	STAA	HIADD	
	LDAA	#\$23	
	STAA	LOADD	
	JSR	READ	read address 123H
LOOP:	BRA	LOOP	wait until reset loop
98C66 *****	FUNCTIONA	L ROUTINES	**************************************
*****	******	******	****************
WRITE	performs	a byte write operat	tion into the 93C66. The routine *
expect	ts the add	ress to modify to h	oe specified in the HIADD and LOADD *
variab	oles. The	new data value is s	specified in the DATVAL variable. *
			specified in the DATVAL variable. * abled state for this function to be *
The 93	3C66 must	be in the write ena	-
The 93	3C66 must ted succes	be in the write end sfully. Each word i	abled state for this function to be *
The 93 execut	3C66 must ted succes igh order	be in the write end sfully. Each word i byte used for ≤odd≤	abled state for this function to be * in the NM93C66 contains 2 bytes with *
The 93 execut the hi	3C66 must ted succes igh order even≤ addr	be in the write end sfully. Each word i byte used for ≤odd≤ esses. The word add	abled state for this function to be * in the NM93C66 contains 2 bytes with * addresses and the low order byte used *
The 93 execut the hi for ≤e dividi	3C66 must ted succes igh order even≤ addr ing the by	be in the write end sfully. Each word i byte used for ≤odd≤ esses. The word add te address specific	abled state for this function to be in the NM93C66 contains 2 bytes with addresses and the low order byte used thresses to be accessed is determined by
The 93 execut the hi for ≤e dividi to be	3C66 must ted succes igh order even≤ addr ing the by written i	be in the write end sfully. Each word is byte used for ≤odd≤ esses. The word add te address specific s determined by rea	abled state for this function to be in the NM93C66 contains 2 bytes with addresses and the low order byte used dresses to be accessed is determined by ted in HIADD and LOADD by two. The data
The 93 execut the hi for ≤e dividi to be mappin	3C66 must ced succes igh order even≤ addr ing the by written i ng in the	be in the write end sfully. Each word i byte used for ≤odd≤ esses. The word add te address specific s determined by reanew byte value into	abled state for this function to be in the NM93C66 contains 2 bytes with addresses and the low order byte used tresses to be accessed is determined by adding the specified word address and *
The 93 execut the hi for ≤e dividi to be mappin	3C66 must ced succes igh order even≤ addr ing the by written i ng in the	be in the write end sfully. Each word i byte used for ≤odd≤ esses. The word add te address specific s determined by reanew byte value into	abled state for this function to be in the NM93C66 contains 2 bytes with addresses and the low order byte used dresses to be accessed is determined by ed in HIADD and LOADD by two. The data adding the specified word address and to the specified high or low order byte.
The 93 execut the hi for ≤e dividi to be mappin	3C66 must ced succes igh order even≤ addr ing the by written i ng in the	be in the write end sfully. Each word if byte used for ≤odd≤ esses. The word add te address specifie s determined by rea new byte value into	abled state for this function to be in the NM93C66 contains 2 bytes with addresses and the low order byte used dresses to be accessed is determined by ed in HIADD and LOADD by two. The data adding the specified word address and to the specified high or low order byte.
The 93 execut the hi for ≤e dividi to be mappin	3C66 must ced succes igh order even≤ addr ing the by written i ng in the ************************************	be in the write end sfully. Each word if byte used for ≤odd≤ esses. The word add te address specifie s determined by rea new byte value into	abled state for this function to be in the NM93C66 contains 2 bytes with addresses and the low order byte used dresses to be accessed is determined by ed in HIADD and LOADD by two. The data adding the specified word address and to the specified high or low order byte.
The 93 execut the hi for ≤e dividi to be mappin	3C66 must ted succes igh order even≤ addr ing the by written i ng in the ************************************	be in the write end sfully. Each word is byte used for <pre></pre> esses. The word add te address specifies a determined by read new byte value into <pre></pre> ************************************	abled state for this function to be in the NM93C66 contains 2 bytes with saddresses and the low order byte used dresses to be accessed is determined by ed in HIADD and LOADD by two. The data adding the specified word address and to the specified high or low order byte. ***********************************
The 93 execut the hi for ≤e dividi to be mappin	3C66 must ced succes igh order even≤ addr ing the by written i ng in the ************************************	be in the write end sfully. Each word is byte used for <pre></pre> esses. The word add te address specifies a determined by read new byte value into <pre></pre> ************************************	abled state for this function to be in the NM93C66 contains 2 bytes with saddresses and the low order byte used dresses to be accessed is determined by ed in HIADD and LOADD by two. The data adding the specified word address and to the specified high or low order byte. ***********************************
The 93 execut the hi for <e be="" dividi="" mappin<="" td="" to=""><td>3C66 must ced succes igh order even≤ addr ing the by written i ng in the ************************************</td><td>be in the write end sfully. Each word if byte used for ≤odd≤ esses. The word add te address specifie s determined by rea new byte value into ************************************</td><td>abled state for this function to be in the NM93C66 contains 2 bytes with saddresses and the low order byte used dresses to be accessed is determined by ed in HIADD and LOADD by two. The data adding the specified word address and to the specified high or low order byte. ***********************************</td></e>	3C66 must ced succes igh order even≤ addr ing the by written i ng in the ************************************	be in the write end sfully. Each word if byte used for ≤odd≤ esses. The word add te address specifie s determined by rea new byte value into ************************************	abled state for this function to be in the NM93C66 contains 2 bytes with saddresses and the low order byte used dresses to be accessed is determined by ed in HIADD and LOADD by two. The data adding the specified word address and to the specified high or low order byte. ***********************************
The 93 execut the hi for ≤e dividi to be mappin	3C66 must ced succes igh order even≤ addr ing the by written i ng in the ************************************	be in the write end sfully. Each word if byte used for ≤odd≤ esses. The word add te address specifie s determined by rea new byte value into ************************************	abled state for this function to be in the NM93C66 contains 2 bytes with saddresses and the low order byte used dresses to be accessed is determined by ed in HIADD and LOADD by two. The data adding the specified word address and to the specified high or low order byte. ***********************************
The 93 execut the hi for ≤e dividi to be mappin	aced must ced succes igh order even≤ addr ing the by written i ng in the ************************************	be in the write end sfully. Each word if byte used for ≤odd≤ esses. The word add te address specifie s determined by rea new byte value into ************************************	abled state for this function to be in the NM93C66 contains 2 bytes with saddresses and the low order byte used dresses to be accessed is determined by ed in HIADD and LOADD by two. The data adding the specified word address and to the specified high or low order byte. ***********************************
The 93 execut the hi for ≤e dividi to be mappin	aced must ced succes igh order even≤ addr ing the by written i ng in the ************************************	be in the write end sfully. Each word if byte used for <pre><pre>SoddS</pre> esses. The word add te address specifies s determined by rea new byte value into ************************************</pre>	abled state for this function to be in the NM93C66 contains 2 bytes with saddresses and the low order byte used dresses to be accessed is determined by ed in HIADD and LOADD by two. The data adding the specified word address and to the specified high or low order byte. ***********************************
The 93 execut the hi for ≤e dividi to be mappin	aced must ced succes igh order even≤ addr ing the by written i ng in the ************************************	be in the write end sfully. Each word if byte used for SoddS esses. The word add te address specifie s determined by rea new byte value into ************************************	abled state for this function to be in the NM93C66 contains 2 bytes with saddresses and the low order byte used dresses to be accessed is determined by ed in HIADD and LOADD by two. The data adding the specified word address and to the specified high or low order byte. ***********************************
The 93 execut the hi for ≤e dividi to be mappin	3C66 must ced succes igh order even≤ addr ing the by written i ng in the ********* LDAA PSHA LDAA	be in the write end sfully. Each word if byte used for SoddS esses. The word add te address specifie s determined by rea new byte value into ************************************	abled state for this function to be in the NM93C66 contains 2 bytes with saddresses and the low order byte used dresses to be accessed is determined by ed in HIADD and LOADD by two. The data adding the specified word address and to the specified high or low order byte. ***********************************
The 93 execut the hi for <e be="" dividi="" mappin<="" td="" to=""><td>aced must ced succes igh order even≤ addr ing the by written i ng in the ************************************</td><td>be in the write end sfully. Each word if byte used for ≤odd≤ esses. The word add te address specifie s determined by rea new byte value into ************************************</td><td>abled state for this function to be in the NM93C66 contains 2 bytes with saddresses and the low order byte used dresses to be accessed is determined by ed in HIADD and LOADD by two. The data adding the specified word address and to the specified high or low order byte. ***********************************</td></e>	aced must ced succes igh order even≤ addr ing the by written i ng in the ************************************	be in the write end sfully. Each word if byte used for ≤odd≤ esses. The word add te address specifie s determined by rea new byte value into ************************************	abled state for this function to be in the NM93C66 contains 2 bytes with saddresses and the low order byte used dresses to be accessed is determined by ed in HIADD and LOADD by two. The data adding the specified word address and to the specified high or low order byte. ***********************************
The 93 execut the hi for <e be="" dividi="" mappin<="" td="" to=""><td>aced must ced succes igh order even≤ addr ing the by written i ng in the ********* LDAA PSHA LD</td><td>be in the write end sfully. Each word if byte used for <pre>SoddSesses. The word add te address specifies s determined by rea new byte value into ************************************</pre></td><td>abled state for this function to be in the NM93C66 contains 2 bytes with saddresses and the low order byte used dresses to be accessed is determined by ed in HIADD and LOADD by two. The data adding the specified word address and to the specified high or low order byte. ***********************************</td></e>	aced must ced succes igh order even≤ addr ing the by written i ng in the ********* LDAA PSHA LD	be in the write end sfully. Each word if byte used for <pre>SoddSesses. The word add te address specifies s determined by rea new byte value into ************************************</pre>	abled state for this function to be in the NM93C66 contains 2 bytes with saddresses and the low order byte used dresses to be accessed is determined by ed in HIADD and LOADD by two. The data adding the specified word address and to the specified high or low order byte. ***********************************
The 93 execut the hi for <e be="" dividi="" mappin<="" td="" to=""><td>3C66 must ced succes igh order even≤ addr ing the by written i ng in the ********* LDAA PSHA STAA JSR PULA STAA PULA STAA</td><td>be in the write end sfully. Each word if byte used for ≤odd≤ esses. The word add te address specifie s determined by rea new byte value into ************************************</td><td>abled state for this function to be in the NM93C66 contains 2 bytes with saddresses and the low order byte used dresses to be accessed is determined by ed in HIADD and LOADD by two. The data adding the specified word address and to the specified high or low order byte. ***********************************</td></e>	3C66 must ced succes igh order even≤ addr ing the by written i ng in the ********* LDAA PSHA STAA JSR PULA STAA PULA STAA	be in the write end sfully. Each word if byte used for ≤odd≤ esses. The word add te address specifie s determined by rea new byte value into ************************************	abled state for this function to be in the NM93C66 contains 2 bytes with saddresses and the low order byte used dresses to be accessed is determined by ed in HIADD and LOADD by two. The data adding the specified word address and to the specified high or low order byte. ***********************************
The 93 execut the hi for ≤e dividi to be mappin	3C66 must ced succes igh order even≤ addr ing the by written i ng in the ******** LDAA PSHA STAA PULA STAA PULA	be in the write end sfully. Each word if byte used for <pre>Sodd</pre> esses. The word add te address specifies s determined by rea new byte value into ************************************	abled state for this function to be in the NM93C66 contains 2 bytes with saddresses and the low order byte used dresses to be accessed is determined by ed in HIADD and LOADD by two. The data adding the specified word address and to the specified high or low order byte. ***********************************
The 93 execut the hi for ≤e dividi to be mappin	aced must ced succes igh order even≤ addr ing the by written i ng in the ********* LDAA PSHA LDAA PSHA LDAA PSHA LDAA PSHA LDAA PSHA EORA STAA JSR PULA STAA PULA STAA PULA STAA	be in the write end sfully. Each word if byte used for <pre>SoddSesses. The word add te address specifies s determined by rea new byte value into ************************************</pre>	abled state for this function to be in the NM93C66 contains 2 bytes with saddresses and the low order byte used dresses to be accessed is determined by ed in HIADD and LOADD by two. The data adding the specified word address and to the specified high or low order byte. ***********************************
The 93 execut the hi for <e be="" dividi="" mappin<="" td="" to=""><td>3C66 must ced succes igh order even≤ addr ing the by written i ng in the ********* LDAA PSHA LDAA PSHA LDAA PSHA LDAA PSHA STAA JSR PULA STAA PULA STAA PULA STAA PULA STAA ROR</td><td>be in the write end sfully. Each word if byte used for ≤odd≤ esses. The word add te address specific s determined by rea new byte value into ************************************</td><td>abled state for this function to be in the NM93C66 contains 2 bytes with saddresses and the low order byte used dresses to be accessed is determined by ed in HIADD and LOADD by two. The data adding the specified word address and to the specified high or low order byte. ***********************************</td></e>	3C66 must ced succes igh order even≤ addr ing the by written i ng in the ********* LDAA PSHA LDAA PSHA LDAA PSHA LDAA PSHA STAA JSR PULA STAA PULA STAA PULA STAA PULA STAA ROR	be in the write end sfully. Each word if byte used for ≤odd≤ esses. The word add te address specific s determined by rea new byte value into ************************************	abled state for this function to be in the NM93C66 contains 2 bytes with saddresses and the low order byte used dresses to be accessed is determined by ed in HIADD and LOADD by two. The data adding the specified word address and to the specified high or low order byte. ***********************************
The 93 execut the hi for ≤e dividi to be mappin	aced must ced succes igh order even≤ addr ing the by written i ng in the ********* LDAA PSHA LDAA PSHA LDAA PSHA LDAA PSHA LDAA PSHA LDAA PSHA STAA JSR PULA STAA PULA STAA PULA STAA PULA STAA PULA STAA ROR ROR	be in the write end sfully. Each word if byte used for <pre> soddsesses. The word add te address specifies s determined by rea new byte value into ************************************</pre>	abled state for this function to be in the NM93C66 contains 2 bytes with saddresses and the low order byte used dresses to be accessed is determined by ed in HIADD and LOADD by two. The data adding the specified word address and to the specified high or low order byte. save the data value save the address determine the byte address in the word that will not be modified read the valid byte retrieve address to modify retrieve new data value calculate word address
The 93 execut the hi for <e be="" dividi="" mappin<="" td="" to=""><td>3C66 must ced succes igh order even≤ addr ing the by written i ng in the ********* LDAA PSHA LDAA PSHA LDAA PSHA LDAA PSHA LDAA PSHA STAA JSR PULA STAA PULA STAA</td><td>be in the write end sfully. Each word if byte used for ≤odd≤ esses. The word add te address specific s determined by rea new byte value into ************************************</td><td>determine the byte address in the word that will not be modified read the valid byte retrieve new data value calculate word address if carry = 0 then we are set</td></e>	3C66 must ced succes igh order even≤ addr ing the by written i ng in the ********* LDAA PSHA LDAA PSHA LDAA PSHA LDAA PSHA LDAA PSHA STAA JSR PULA STAA	be in the write end sfully. Each word if byte used for ≤odd≤ esses. The word add te address specific s determined by rea new byte value into ************************************	determine the byte address in the word that will not be modified read the valid byte retrieve new data value calculate word address if carry = 0 then we are set
The 93 execut the hi for <e be="" dividi="" mappin<="" td="" to=""><td>aced must ced succes igh order even≤ addr ing the by written i ng in the ********* LDAA PSHA LDAA PSHA LDAA PSHA LDAA PSHA LDAA PSHA LDAA PSHA STAA JSR PULA STAA PULA STAA PULA STAA PULA STAA PULA STAA ROR ROR</td><td>be in the write end sfully. Each word if byte used for <pre> soddsesses. The word add te address specifies s determined by rea new byte value into ************************************</pre></td><td>abled state for this function to be in the NM93C66 contains 2 bytes with saddresses and the low order byte used dresses to be accessed is determined by ed in HIADD and LOADD by two. The data adding the specified word address and to the specified high or low order byte. save the data value save the address determine the byte address in the word that will not be modified read the valid byte retrieve address to modify retrieve new data value calculate word address</td></e>	aced must ced succes igh order even≤ addr ing the by written i ng in the ********* LDAA PSHA LDAA PSHA LDAA PSHA LDAA PSHA LDAA PSHA LDAA PSHA STAA JSR PULA STAA PULA STAA PULA STAA PULA STAA PULA STAA ROR ROR	be in the write end sfully. Each word if byte used for <pre> soddsesses. The word add te address specifies s determined by rea new byte value into ************************************</pre>	abled state for this function to be in the NM93C66 contains 2 bytes with saddresses and the low order byte used dresses to be accessed is determined by ed in HIADD and LOADD by two. The data adding the specified word address and to the specified high or low order byte. save the data value save the address determine the byte address in the word that will not be modified read the valid byte retrieve address to modify retrieve new data value calculate word address
The 93 execut the hi for ≤e dividi to be mappin	3C66 must ced succes igh order even≤ addr ing the by written i ng in the ********* LDAA PSHA LDAA PSHA LDAA PSHA LDAA PSHA LDAA PSHA STAA JSR PULA STAA	be in the write end sfully. Each word if byte used for <pre> soddsesses. The word add te address specifies s determined by rea new byte value into ************************************</pre>	determine the byte address in the word that will not be modified read the valid byte save the address to modify retrieve new data value calculate word address if carry = 0 then we are set
The 93 execut the hi for ≤e dividi to be mappin	3C66 must ced succes igh order even≤ addr ing the by written i ng in the ********* LDAA PSHA LDAA PSHA LDAA PSHA LDAA PSHA STAA JSR PULA STAA	be in the write end sfully. Each word if byte used for <pre>SoddSesses. The word add te address specifies s determined by rea new byte value into ************************************</pre>	determine the byte address in the word that will not be modified read the valid byte retrieve new data value calculate word address if carry = 0 then we are set save new data value

	STAA	DATVAL		into DATVAL
NOFLIP:	BSET	PORTD, X	#CSBIT	enable device
	LDAA	#\$05		
	JSR	SENDB		send WRITE instruction
	LDAA	LOADD		
	JSR	SENDB		send word address to modify
	LDAA	DATVAL		
	JSR	SENDB		send high order data byte
	LDAA	TEMP		
	JSR	SENDB		send low order data byte
	BCLR	PORTD, X	#CSBIT	disable device
	JSR RTS	BUSY		wait until write has completed
	KIS			

_				le 93000. The foucine
_			_	I III CHE HIADD AND LOADD
		_		ss is recurred in the
				contains 2 bytes with the
	_			id the low order byte used
				be accessed is determined
_		_		HIADD and LOADD by two. ************************************
^ ~ ~ ~ * * * *	^ * * * * * * * *	^ ^ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~	^ ^ * * * * * * * * * * *	, , , , , , , , , , , , , , , , , , ,
DD3.5.	Danm	DODED ***	# GOD TH	anabla danina
READ:	BSET	PORTD, X	#CSBIT	enable device
READ:	LDAA	#\$06	#CSBIT	
READ:	LDAA JSR	#\$06 SENDB	#CSBIT	enable device send READ instruction
READ:	LDAA JSR ROR	#\$06 SENDB HIADD	#CSBIT	
READ:	LDAA JSR ROR ROR	#\$06 SENDB HIADD LOADD	#CSBIT	
READ:	LDAA JSR ROR ROR LDAA	#\$06 SENDB HIADD LOADD LOADD	#CSBIT	send READ instruction
READ:	LDAA JSR ROR ROR LDAA JSR	#\$06 SENDB HIADD LOADD LOADD SENDB	#CSBIT	
READ:	LDAA JSR ROR ROR LDAA JSR LDAA	#\$06 SENDB HIADD LOADD LOADD SENDB #\$54	#CSBIT	send READ instruction
READ:	LDAA JSR ROR ROR LDAA JSR LDAA STAA	#\$06 SENDB HIADD LOADD LOADD SENDB #\$54 SPCR, X	#CSBIT	send READ instruction send address sample on falling edge when reading
READ:	LDAA JSR ROR ROR LDAA JSR LDAA STAA JSR	#\$06 SENDB HIADD LOADD LOADD SENDB #\$54 SPCR, X SENDB	#CSBIT	send READ instruction
READ:	LDAA JSR ROR ROR LDAA JSR LDAA STAA JSR BCS	#\$06 SENDB HIADD LOADD LOADD SENDB #\$54 SPCR, X SENDB DONER	#CSBIT	send READ instruction send address sample on falling edge when reading
	LDAA JSR ROR ROR LDAA JSR LDAA STAA JSR BCS JSR	#\$06 SENDB HIADD LOADD LOADD SENDB #\$54 SPCR, X SENDB DONER SENDB	#CSBIT	send READ instruction send address sample on falling edge when reading
	LDAA JSR ROR ROR LDAA JSR LDAA STAA JSR BCS JSR STAA	#\$06 SENDB HIADD LOADD LOADD SENDB #\$54 SPCR, X SENDB DONER SENDB		send READ instruction send address sample on falling edge when reading read data value
	LDAA JSR ROR ROR LDAA JSR LDAA STAA JSR BCS JSR STAA BCLR	#\$06 SENDB HIADD LOADD LOADD SENDB #\$54 SPCR, X SENDB DONER SENDB DATVAL PORTD, X	#CSBIT	send READ instruction send address sample on falling edge when reading
	LDAA JSR ROR LDAA JSR LDAA STAA JSR BCS JSR STAA BCLR LDAA	#\$06 SENDB HIADD LOADD LOADD SENDB #\$54 SPCR, X SENDB DONER SENDB DATVAL PORTD, X #\$50	#CSBIT	send READ instruction send address sample on falling edge when reading read data value disable device
	LDAA JSR ROR ROR LDAA JSR LDAA STAA JSR BCS JSR STAA BCLR LDAA STAA	#\$06 SENDB HIADD LOADD LOADD SENDB #\$54 SPCR, X SENDB DONER SENDB DATVAL PORTD, X	#CSBIT	send READ instruction send address sample on falling edge when reading read data value
DONER:	LDAA JSR ROR ROR LDAA JSR LDAA STAA JSR BCS JSR STAA BCLR LDAA STAA RTS	#\$06 SENDB HIADD LOADD LOADD SENDB #\$54 SPCR, X SENDB DONER SENDB DATVAL PORTD, X #\$50 SPCR, X	#CSBIT data val:	send READ instruction send address sample on falling edge when reading read data value disable device id on rising edge
DONER:	LDAA JSR ROR LDAA JSR LDAA STAA JSR BCS JSR STAA BCLR LDAA STAA RTS	#\$06 SENDB HIADD LOADD LOADD SENDB #\$54 SPCR, X SENDB DONER SENDB DATVAL PORTD, X #\$50 SPCR, X	#CSBIT data val:	send READ instruction send address sample on falling edge when reading read data value disable device id on rising edge
DONER: ******* * WEN en	LDAA JSR ROR ROR LDAA JSR LDAA STAA JSR BCS JSR STAA BCLR LDAA STAA BCLR LDAA STAA RTS ***********************************	#\$06 SENDB HIADD LOADD LOADD SENDB #\$54 SPCR, X SENDB DONER SENDB DATVAL PORTD, X #\$50 SPCR, X	#CSBIT data val: ***********************************	send READ instruction send address sample on falling edge when reading read data value disable device id on rising edge
DONER: ******* * WEN en * along	LDAA JSR ROR ROR LDAA JSR LDAA STAA JSR BCS JSR STAA BCLR LDAA STAA RTS ********	#\$06 SENDB HIADD LOADD LOADD SENDB #\$54 SPCR, X SENDB DONER SENDB DATVAL PORTD, X #\$50 SPCR, X	#CSBIT data val: ***********************************	send READ instruction send address sample on falling edge when reading read data value disable device id on rising edge **********************************
DONER: ****** * WEN en * along * inadve	LDAA JSR ROR ROR LDAA JSR LDAA STAA JSR BCS JSR STAA BCLR LDAA STAA RTS ***********************************	#\$06 SENDB HIADD LOADD LOADD SENDB #\$54 SPCR, X SENDB DONER SENDB DATVAL PORTD, X #\$50 SPCR, X	#CSBIT data val: ************* rm a write op ble) function	send READ instruction send address sample on falling edge when reading read data value disable device id on rising edge **********************************
DONER: ******* * WEN en * along * inadve	LDAA JSR ROR ROR LDAA JSR LDAA STAA JSR BCS JSR STAA BCLR LDAA STAA RTS ***********************************	#\$06 SENDB HIADD LOADD LOADD SENDB #\$54 SPCR, X SENDB DONER SENDB DATVAL PORTD, X #\$50 SPCR, X	#CSBIT data val: ************* rm a write op ble) function	send READ instruction send address sample on falling edge when reading read data value disable device id on rising edge **********************************
******* * WEN en * along * inadve *****	LDAA JSR ROR ROR LDAA JSR LDAA STAA JSR BCS JSR STAA BCLR LDAA STAA RTS ***********************************	#\$06 SENDB HIADD LOADD LOADD SENDB #\$54 SPCR, X SENDB DONER SENDB DATVAL PORTD, X #\$50 SPCR, X	#CSBIT data val: ************* rm a write op ble) function	send READ instruction send address sample on falling edge when reading read data value disable device id on rising edge **********************************
******* * WEN en * along * inadve ******	LDAA JSR ROR ROR LDAA JSR LDAA STAA JSR BCS JSR STAA BCLR LDAA STAA RTS ***********************************	#\$06 SENDB HIADD LOADD LOADD SENDB #\$54 SPCR, X SENDB DONER SENDB DATVAL PORTD, X #\$50 SPCR, X	#CSBIT data val: ***********************************	send READ instruction send address sample on falling edge when reading read data value disable device id on rising edge **********************************
******* * WEN en * along * inadve ******	LDAA JSR ROR ROR LDAA JSR LDAA STAA JSR BCS JSR STAA BCLR LDAA STAA RTS ***********************************	#\$06 SENDB HIADD LOADD LOADD SENDB #\$54 SPCR, X SENDB DONER SENDB DATVAL PORTD, X #\$50 SPCR, X	#CSBIT data val: ***********************************	send READ instruction send address sample on falling edge when reading read data value disable device id on rising edge **********************************
******* * WEN en * along * inadve ******	LDAA JSR ROR ROR LDAA JSR LDAA STAA JSR BCS JSR STAA BCLR LDAA STAA RTS ***********************************	#\$06 SENDB HIADD LOADD LOADD SENDB #\$54 SPCR, X SENDB DONER SENDB DATVAL PORTD, X #\$50 SPCR, X ***********************************	#CSBIT data val: ***********************************	send READ instruction send address sample on falling edge when reading read data value disable device id on rising edge **********************************
DONER: ******* * WEN en * along * inadve *******	LDAA JSR ROR ROR LDAA JSR LDAA STAA JSR BCS JSR STAA BCLR LDAA STAA RTS ********* LDAA RTS ********* BSET LDAA JSR	#\$06 SENDB HIADD LOADD LOADD SENDB #\$54 SPCR, X SENDB DONER SENDB DATVAL PORTD, X #\$50 SPCR, X ***********************************	#CSBIT data val: ***********************************	send READ instruction send address sample on falling edge when reading read data value disable device id on rising edge **********************************
* WEN en * along * inadve	LDAA JSR ROR ROR LDAA JSR LDAA STAA JSR BCS JSR STAA BCLR LDAA STAA RTS ******** ables the with the stant dat ******** BSET LDAA JSR LDAA JSR LDAA	#\$06 SENDB HIADD LOADD LOADD SENDB #\$54 SPCR, X SENDB DONER SENDB DATVAL PORTD, X #\$50 SPCR, X ***********************************	#CSBIT data val: ***********************************	send READ instruction send address sample on falling edge when reading read data value disable device id on rising edge **********************************

* funct	ion preven	ts against inadv	vertant data	changes.	*
*****	******	******	*****	**********	***
WDS:	BSET	PORTD, X	#CSBIT	enable device	
	LDAA	#\$04			
	JSR	SENDB		send WDS instruction	
	LDAA	#\$00			
	JSR	SENDB		send instruction + dummy address	
	BCLR	PORTD, X	#CSBIT	disable device	
	RTS				
*****	*****	* * * * * * * * * * * * * * * * *	*****	**********	***
* SUPPO	RT ROUTINE	S			,
*****	*****	******	*****	***********	***
*****	*****	******	******	**********	***

* BUSY	is used to	pause until a v	write or eras	e operation has completed.	7
* BUSY ****** BUSY:	is used to ******* BSET	pause until a v ***********************************	write or eras	e operation has completed. ***********************************	•
* BUSY	is used to ****** BSET BRCLR	pause until a v **************** PORTD, X PORTD, X	#CSBIT #\$04	e operation has completed. ***********************************	7
* BUSY ****** BUSY: TWC:	is used to ******* BSET	pause until a v ***********************************	write or eras	e operation has completed. ***********************************	4
* BUSY ****** BUSY:	is used to ****** BSET BRCLR	pause until a v **************** PORTD, X PORTD, X	#CSBIT #\$04	e operation has completed. ***********************************	7
* BUSY ****** BUSY: TWC:	is used to ******** BSET BRCLR BCLR	pause until a v *********** PORTD, X PORTD, X PORTD, X	write or eras ***** #CSBIT #\$04 #CSBIT	e operation has completed. ***********************************	* * * :
* BUSY ****** BUSY: TWC: RTS ******	is used to ******** BSET BRCLR BCLR	pause until a v ************ PORTD, X PORTD, X PORTD, X	write or eras ****** #CSBIT #\$04 #CSBIT	e operation has completed. ***********************************	***
* BUSY ****** BUSY: TWC: RTS ****** * SENDB	is used to ******* BSET BRCLR BCLR ********** is used to	pause until a v ************ PORTD, X PORTD, X PORTD, X	#CSBIT #\$04 #CSBIT	e operation has completed. ***********************************	***
* BUSY ****** BUSY: TWC: RTS ****** * SENDB	is used to ******* BSET BRCLR BCLR ********* is used to has been re	pause until a v ************* PORTD, X PORTD, X PORTD, X o send a byte to eturned to the 6	#CSBIT #\$04 #CSBIT #05 #CSBIT	e operation has completed. ***********************************	***
* BUSY ****** BUSY: TWC: RTS ****** * SENDB	is used to ******* BSET BRCLR BCLR ********* is used to has been re	pause until a v ************* PORTD, X PORTD, X PORTD, X o send a byte to eturned to the 6	#CSBIT #\$04 #CSBIT #05 #CSBIT	e operation has completed. ***********************************	***
* BUSY ****** BUSY: TWC: RTS ****** * SENDB * that ****** SENDB:	is used to ******* BSET BRCLR BCLR ******** is used thas been re **********	pause until a v ************ PORTD, X PORTD, X PORTD, X ***********************************	#CSBIT #\$04 #CSBIT ************************************	e operation has completed. ***********************************	· · · · · · · · · · · · · · · · · · ·
* BUSY ****** BUSY: TWC: RTS ****** * SENDB * that ******	is used to ******* BSET BRCLR BCLR ******** is used to	pause until a v ************ PORTD, X PORTD, X PORTD, X ***************** o send a byte to eturned to the 6 ************************************	#CSBIT #\$04 #CSBIT ************************************	e operation has completed. ***********************************	***

Life Support Policy

Fairchild's products are not authorized for use as critical components in life support devices or systems without the express written approval of the President of Fairchild Semiconductor Corporation. As used herein:

- 1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
- 2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

Fairchild Semiconductor Americas	Fairchild Sem Europe	iconductor	Fairchild Semiconductor Hong Kong	Fairchild Semiconductor Japan Ltd.
Customer Response Center	Fa	x: +44 (0) 1793-856858	8/F, Room 808, Empire Centre	4F, Natsume Bldg.
Tel. 1-888-522-5372	Deutsch Te	el: +49 (0) 8141-6102-0	68 Mody Road, Tsimshatsui East	2-18-6, Yushima, Bunkyo-ku
	English Te	el: +44 (0) 1793-856856	Kowloon, Hong Kong	Tokyo, 113-0034 Japan
	Français Te	el: +33 (0) 1-6930-3696	Tel; +852-2722-8338	Tel: 81-3-3818-8840
	Italiano Te	el: +39 (0) 2-249111-1	Fax: +852-2722-8383	Fax: 81-3-3818-8841