

I²C two wire communication

Communicating with the I²C bus using the GMS87C1202

DESCRIPTION

I²C serial EEPROMs feature a two wire serial interface bus. The bus protocol is I²C compatible. Interface to a serial port with I²C bus protocol in a microcontroller is trivial. This application note is intended for design engineers who want to develop their software programs to communicate a microcontroller with a 2-wire bus Serial EEPROM through a general purpose I/O port.

Unlike the 3-wire bus Serial EEPROMs, the X24CXX communicate with any microcontroller only by a serial data I/O line (SDA) and a serial clock (SCL). Chip select is not required. Data transfer may be initiated only when the bus is not busy. During such transfer, the data line (SDA) must remain stable whenever the clock line (SCL)

is high. Changes in the data line while the clock line is high are interpreted as a START or STOP condition. A typical transfer format is shown in Figure 1.

After the START condition, a slave address is sent. This address is 7-bits long, the eighth bit is a data direction bit. (R/W - a logical '0' indicates a transmission WRITE, a logical '1' represents a request for data READ. A data transfer is always terminated by a STOP condition generated by the master controller. However, if a master still wishes to communicate on the bus, it can generate another START condition and address another slave without first generating a STOP condition. Various combinations of read/write formats are then possible within such transfer.

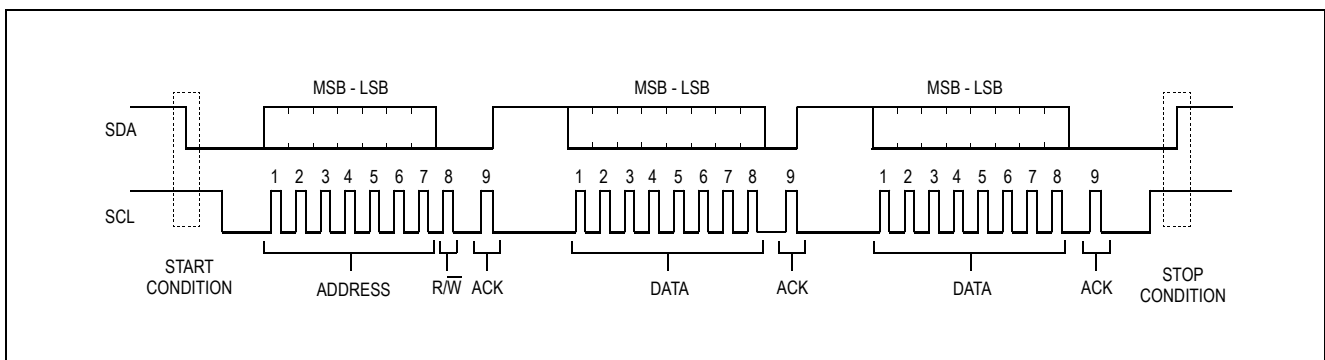


Figure 1. Transfer format

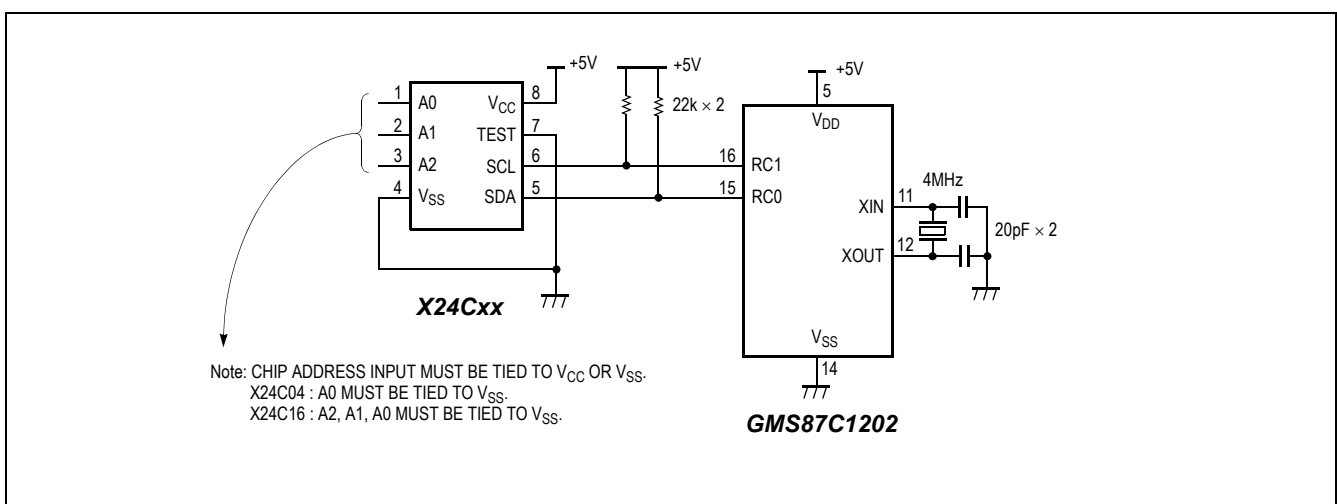


Figure 2. A simple hardware connection

An example program has been provided in Appendix A containing all GMS87C1202 routines needed to exercise a X24C02 device. A simple hardware connection is illustrated in Figure 2. A maximum of eight X24C02, or four X24C04's can be addressed by a microcontroller on the same two wire bus without additional interfaces. Each device is identified by its Chip Address and will only respond to the correct slave address. A detailed bus flow is shown in Figure 3. Figure 3 as shown below describes how the bit stream is set up for READ and WRITE mode in the microcomputer programming software prior to

sending it on the two wire serial bus.

The stop condition, after the one byte data write sequence, starts the internal self-timed write cycle which may last up to 0.56 milliseconds (start+150µs per byte). Acknowledge signal should be monitored during this period....

Author: **Jongpil Shin**
MCU application team
e-mail: jpshin@hynix.com

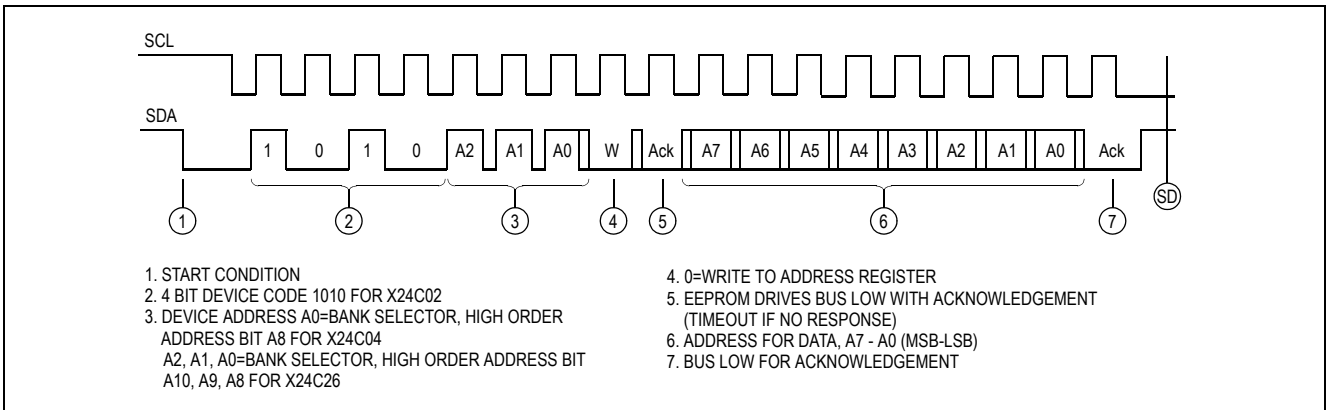


Figure 3. Setting the internal word address of the X24CXX

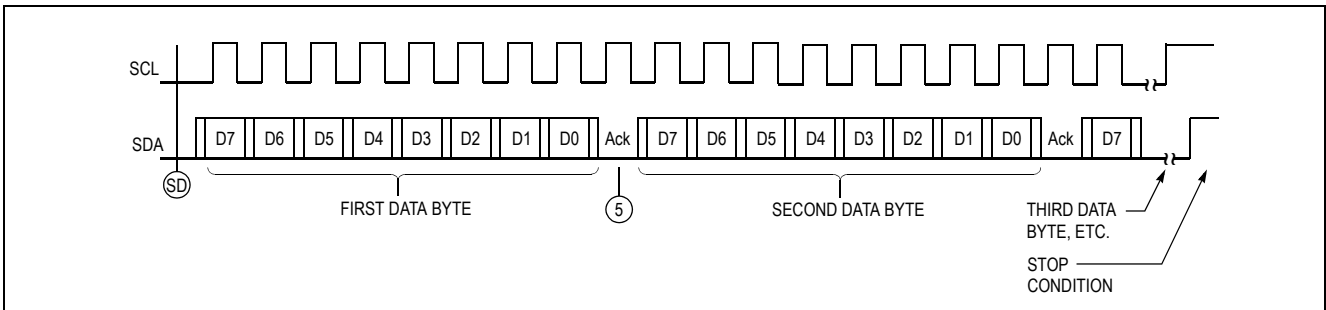


Figure 4. Byte Write Sequence

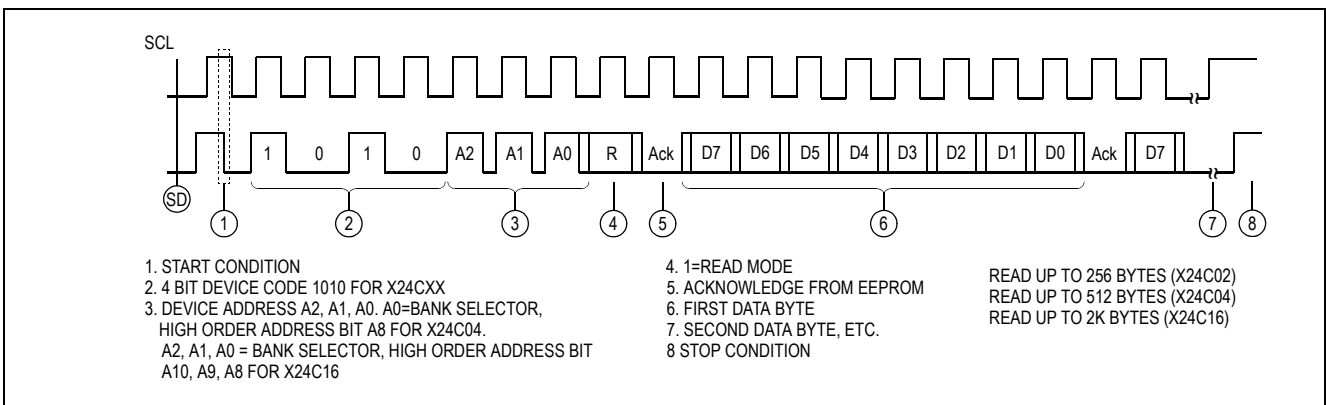


Figure 5. Read Mode Sequence

Appendix A:

GMS800 series MICOM ASSEMBLER Fri Aug 20 13:14:24 1999
(PAGE 1)

```

1          PAGE      10000
2          ;*****
3          ; PROGRAM DESCRIPTION:
4          ; Many system are implemented by IIC communication theory, it is came from
5          ; Philips and use two wires, one is clock line and another is data line.
6          ;-----
7          ; MICROCONTROLLER : GMS87C1202 (2K ROM, 20 PIN DIP)
8          ; OPER. FREQUENCY : 4MHz
9          ; SERIAL EEPROM   : X24C02 (2KBIT, 8 PIN DIP)
10         ;-----
11         ; programmed by Jongpil Shin (HEI) on 1999.08.20      Version 1.00
12         ;*****
13         ;
14         ;          TYPICAL APPLICATION CIRCUIT
15         ;
16         ;          +-----+
17         ;          |          ( )          |          +5V
18         ;          1|RA4/AN4      AN3/RA3|20      |
19         ;          |          |          |          +-----+
20         ;          2|RA5/AN5      AN2/RA2|19      | | |
21         ;          |          |          |          | | |
22         ;          3|RA6/AN6 G AN1/RA1|18      / /22K |
23         ;          |          M          |          / / | 7+-----+
24         ;          4|RA7/AN7 S EC0/RA0|17      | | +--|VDD WP|--+
25         ;          |          8          |          | | 6| | |
26         ;          +5V 5|VDD      7      RC1|16-----+-----|SCL A2|--+
27         ;          |          C          |          | | 5| | |
28         ;          6|RB0/AN0 1      RC0|15-----+-----|SDA A1|--+
29         ;          |          2          |          | | |
30         ;          7|RB1/BUZ 0      VSS|14 GND      | | A0|--+
31         ;          |          2          |          | | |
32         ;          8|RB2/INT0 /RESET|13 /Reset      | | VSS|--+
33         ;          |          |          |          +-----+ |
34         ;          9|RB3/INT1      XOUT|12 X-tal Out      24C02 GND
35         ;          |          |          |          EEPROM
36         ;          10|RB4/PWM/COMP XIN|11 X-tal In
37         ;          |          |
38         ;          +-----+
39         ;          GMS87C1202
40         ;
41         ;*****
42         ; * Included Subroutines
43         ; IIC_Rx : Receive one byte through the IIC bus
44         ; IIC_Tx : Transmmit one byte through the IIC bus
45         ; IIC_start : Goes to "start condition"
46         ; IIC_stop : Goes to "stop condition"
47         ;*****
48         ; * Using I/O pins
49         ; PIO_SDA : IIC bus serial data line port, the RC0 pin is used
50         ; PO_SCL : IIC bus serial clock line port, the RC1 pin is used
51         ;*****
52         ; * Used RAM
53         ; - none
54         ;*****
55         ;
56         RC EQU 0C4H
57         RCIO EQU 0C5H
58

```

```

59     INPUT_MODE EQU    0000_0010B    ; make SDA pin as an input port to read acknowledge
60     OUTPUT_MODE EQU   0000_0011B    ; make SDA pin as an output port to write bit and acknowledge
61     PO_SCL EQU       1,RC
62     PIO_SDA EQU      0,RC
63
64     ;***** VECTOR AREA *****
65     ;
66             ORG      0FFFEH
67 FFFE 00F8      DW      RESET          ;Reset Vector
68
69             ORG      0F800H
70
71 F800 1E7F  RESET:  LDX   #07FH          ;Initialize SP = #7FH
72 F802 8E      TXSP
73 F803 E403C4  LDM   RC,#0000_0011B
74 F806 E403C5  LDM   RCIO,#0000_0011B
75 F809 1BCBF8  JMP    START
76
77     ;*****
78     ; Write one byte data into the EEPROM
79     ; - Write one byte data (Accumulator) at address X of EEPROM
80     ;*****
81     ; Input parameter
82     ; - Accumulator : write data
83     ; - X register  : write address
84     ; Output parameter
85     ; - None
86     ; Scatched data
87     ; - Y register  : shift bit counter
88     ;-----
89     ; This write cycle time is the time(max10ms) from a valid stop condition of a
90     ; write sequence to the end of the internal erase/program cycle, During the write
91     ; cycle, the EEPROM bus interface circuits are disabled, SDA is allowed to
92     ; remain high, and the device does not respond to its slave address.
93     ;*****
94
95     write_byte:
96 F80C 0E      PUSH   A
97 F80D 3B96F8  CALL   IIC_start
98 F810 C4A0    LDA    #0A0H      ; set to be "byte write condition"
99 F812 3B3DF8  CALL   IIC_Tx      ; tx. one byte via IIC bus
100 F815 EE     XAX          ; set word address (X -> A)
101 F816 3B3DF8  CALL   IIC_Tx      ; tx. one byte(address) to slave
102 F819 0D     POP    A          ; set one byte data to be wrote
103 F81A 3B3DF8  CALL   IIC_Tx      ; tx. one byte(data) to slave
104 F81D 3BA7F8  CALL   IIC_stop    ; make "stop condition"
105 F820 6F     RET
106
107     ;*****
108     ; Read one byte data from the EEPROM
109     ; - One byte data is read from EEPROM address X of EEPROM
110     ;*****
111     ; Input parameter
112     ; - X register  : read address
113     ; Output parameter
114     ; - Accumulator : one byte data
115     ; Scatched data
116     ; - Y register  : shift bit counter
117     ;*****
118     ReadFromEEPROM:
119     read_byte:
120 F821 3B96F8  CALL   IIC_start

```

```

121 F824 C4A0          LDA    #0A0H          ; set to be "byte write condition"
122 F826 3B3DF8       CALL   IIC_Tx          ; transmit one byte (slave address) via IIC bus
123 F829 EE           XAX                    ; set word address (X -> A)
124 F82A 3B3DF8       CALL   IIC_Tx          ; transmit one byte(word address) to slave
125 F82D 3B96F8       CALL   IIC_start      ; set to be start condition
126 F830 C4A1          LDA    #0A1H          ; set to be "byte write condition"
127 F832 3B3DF8       CALL   IIC_Tx          ; transmit one byte (slave address) via IIC bus
128 F835 FF           NOP
129 F836 3B7BF8       CALL   IIC_Rx         ; receive one byte(data) to slave
130 F839 3BA7F8       CALL   IIC_stop       ; make "stop condition"
131 F83C 6F           RET
132                   ;
133                   ;-----
134                   ; TRANSMIT ONE BYTE DATA VIA IIC BUS
135                   ;-----
136                   ; Entry      : Accumulator (to be wrote transmit data)
137                   ; Output     : Carry (0=ok, 1=No ack, maybe error)
138                   ; Scratch data : Y register is used shift counter and will be decremented
139                   ;-----
140 F83D E403C5       IIC_Tx: LDM    RCIO,#OUTPUT_MODE ; set SDA as an output mode
141
142 F840 3E06          LDY    #6              ; prepare shift counter seven clock counter
143                   iic_loop: ; Transmit seven clocks
144 F842 28           ROL    A              ; get one bit to be send
145 F843 EBC410       STC    PIO_SDA        ; transmit one bit through the data line
146 F846 FF           NOP
147 F847 FF           NOP
148 F848 FF           NOP
149 F849 21C4         SET1   PO_SCL         ; make clock rising edge
150 F84B FF           NOP
151 F84C FF           NOP
152 F84D 31C4         CLR1   PO_SCL        ; make clock falling edge
153 F84F BE           DEC    Y              ; eight bits done ?
154 F850 10F0         BPL   iic_loop       ; no -> coutinue tx. the bit
155                   ; Transmit eighth clock
156                   ; - Ack. form slave is output when eighth clock is fall, so SDA pin have to be input mode
157                   ; as below
158 F852 FF           NOP
159 F853 28           ROL    A
160 F854 EBC410       STC    PIO_SDA
161 F857 5009         BCC   iic_l3         ; LSB is 0 ?
162 F859 21C4         SET1   PO_SCL        ; No, LSB is "1", SDA have to be input mode before fall...
163 F85B E402C5       LDM    RCIO,#INPUT_MODE ; ... of eighth clock to prevent skew with Ack. from slave
164 F85E 31C4         CLR1   PO_SCL
165 F860 2F0C         BRA   check_ack
166 F862 21C4         iic_l3: SET1   PO_SCL ; Yes, LSB is "0", there is no ban skew condition between SDA and
Ack.
167 F864 FF           NOP ; SDA goes to be input mode normally after fall of eighth clock
168 F865 FF           NOP
169 F866 FF           NOP
170 F867 FF           NOP
171 F868 FF           NOP
172 F869 31C4         CLR1   PO_SCL
173 F86B E402C5       LDM    RCIO,#INPUT_MODE ; make SDA port as an input port to read ack. signal
174                   ; all eight bits data is transmitted,
175                   ; this time acknowledge bit have to be checked
176                   check_ack: ; Transmit ninth clock
177 F86E 3BBEF8       CALL   delay1
178 F871 21C4         SET1   PO_SCL        ; Ninth clock cycle output to read Ack.
179 F873 CBC410       LDCB   PIO_SDA      ; Read Ack. If Carry is 1, then error
180 F876 31C4         CLR1   PO_SCL        ; for next procedure
181 F878 01C4         SET1   PIO_SDA      ; make to be initial state...

```

```

182 F87A 6F          RET
183                ;-----
184                ; RECEIVE ONE BYTE DATA VIA IIC BUS
185                ;-----
186                ; Entry      : None
187                ; Output     : Accumulator (Received one byte data)
188                ; Scratch data : Y will be decremented
189                ;-----
190 F87B 3E07      IIC_Rx: LDY      #7          ; prepare shift counter
191 F87D FF        iic_l2: NOP
192 F87E FF        NOP
193 F87F 21C4      SET1     PO_SCL      ; request to slave (EEPROM) output one bit
194 F881 CBC400    LDC      PIO_SDA     ; Read one bit and store to Carry
195 F884 28        ROL      A
196 F885 31C4      CLR1     PO_SCL      ; end read one bit
197 F887 BE        DEC      Y          ; all eight bits done ?
198 F888 10F3      BPL      iic_l2     ; no -> read rx. data coutinuously
199                ;
200                ; It should be noted that the ninth clock cycle of the read operation is not a "don't care".
201                ; To terminate a read operation, the master must either issue a stop condition during the
202                ; ninth cycle or hold SDA HIGH during the ninth clock cycle and then issue a stop condition.
203 F88A E403C5    LDM      RCIO,#OUTPUT_MODE ; make SDA port as an input port to read ack. signal
204 F88D 01C4      SET1     PIO_SDA     ; hold SDA is "high" during ninth clock
205 F88F FF        NOP
206 F890 21C4      SET1     PO_SCL      ; generate ninth clock
207 F892 FF        NOP                ; delay
208 F893 31C4      CLR1     PO_SCL      ; end session
209 F895 6F        RET
210                ;-----
211                ; Make EEPROM to be "START CONDITION"
212                ; - All commands are preceded by the start condition which
213                ;   is a HIGH to LOW transition of SDA when SCL is HIGH.
214                ;-----
215 IIC_start:
216 F896 E403C5    LDM      RCIO,#OUTPUT_MODE ; set SDA as an output mode
217 F899 01C4      SET1     PIO_SDA     ; initialize SDA to be normal state
218 F89B 21C4      SET1     PO_SCL      ; initialize SDA to be normal state
219 F89D 3BB5F8    CALL     delay4
220 F8A0 11C4      CLR1     PIO_SDA     ; make start condition throught execution below line
221 F8A2 FF        NOP
222 F8A3 FF        NOP
223 F8A4 31C4      CLR1     PO_SCL
224 F8A6 6F        RET
225                ;-----
226                ; Make EEPROM to be "STOP CONDITION"
227                ; - All communication must be terminated by a sotp condition,
228                ;   which is a LOW to High transition of SDA when SCL is HIGH.
229                ;-----
230 F8A7 E403C5    IIC_stop: LDM      RCIO,#OUTPUT_MODE ; set SDA as an output mode
231 F8AA 11C4      CLR1     PIO_SDA
232 F8AC FF        NOP
233 F8AD FF        NOP
234 F8AE 21C4      SET1     PO_SCL
235 F8B0 FF        NOP
236 F8B1 FF        NOP
237 F8B2 01C4      SET1     PIO_SDA
238 F8B4 6F        RET
239
240                ;-----
241                ; Delay routine
242                ; - delay1 : 13 cycles long, 13*0.5us = 6.5us
243                ; - delay2 : 26 cycles long, 26*0.5us = 13.0us

```

```

244          ; - delay4 : 52 cycles long, 52*0.5us = 26.0us
245          ; - delay4 : 104 cycles long, 104*0.5us = 52.0us
246          ;-----
247
248 F8B5 3BB8F8 delay4: call    delay3
249 F8B8 3BBBF8 delay3: call    delay2
250 F8BB 3BBEF8 delay2: call    delay1
251 F8BE 6F      delay1: RET                ; execution time needs 5 cycle
252
253          delay10ms:                ; approximately 10ms
254 F8BF 1E0E          LDX    #14
255 F8C1 3EED          dx1:  LDY    #0EDH
256 F8C3 BE            dy1:  DEC    Y
257 F8C4 70FD          BNE    dy1
258 F8C6 AF            DEC    X
259 F8C7 70F8          BNE    dx1
260 F8C9 FF            NOP
261 F8CA 6F            RET
262
263          ;Test program to read and write random
264 F8CB 1E03          START: LDX    #3          ; set r/w loc. = 2
265 F8CD C455          LDA    #85          ; write 85
266 F8CF 3B0CF8          CALL   write_byte  ; write 85 to EEPROM
267 F8D2 3BBFF8          CALL   delay10ms  ; delay 10ms for write processing
268 F8D5 1E03          LDX    #3
269 F8D7 3B21F8          CALL   read_byte
270 F8DA 4455          CMP    #85
271 F8DC F003          BEQ    CORRECT
272          WRONG:
273 F8DE 1BDEF8          JMP    WRONG
274          CORRECT:
275 F8E1 1BE1F8          JMP    CORRECT
276
277          END

```

-- 0 Error(s) --

--- Total Machine Code : 230 Bytes --