PCM-9361 SMBus/GPIO Programming Note

ICs: Intel ICH7M
PCA9554

Base I/O address----------500h

ICH6 SMBus I/O Registers offset:

Host Status Register ----------00h

Bit 7 (DS) default → 0

0 *= After a full PCI reset, a read to this bit returns a 0.*

Bit 6 (INUSE_STS) default → 0

0 *= Interrupt or SMI# was not generated by SMBALERT#. This bit is only cleared by software writing a 1 to the bit position or by RSMRST# going low.*

Bit 5 (SMBALERT_STS) default → 0

0 *= Cleared by writing a 1 to the bit position.*

Bit 4 (FAILED) default → 0

0 *= Cleared by writing a 1 to the bit position.*

Bit 3 (BUS_ERR) default → 0

0 *= Cleared by writing a 1 to the bit position.*

Bit 2 (DEV_ERR) default → 0

0 *= Software resets this bit by writing a 1 to this location. The ICH4 will then deassert the interrupt or SMI#.*

Bit 1 (INTR) default → 0

0 *= Software resets this bit by writing 1 to this location. The ICH4 will then deassert the interrupt or SMI#.*

1 *= The source of the interrupt or SMI# was the successful completion of its last command.*

Bit 0 (HOST_BUSY) default → 0

Host Control Register ----------02h

Bit 7 (PEC_EN) default → 0

0 *= SMBus host controller does not perform the transaction with the PEC phase appended.*

Bit 6 (START) default → 0

0 *= This bit will always return 0 on reads. The HOST_BUSY bit in the Host Status register (offset 00h) can be used to identify when the ICH4 has finished the command.*

1 *= Writing a 1 to this bit initiates the command described in the SMB_CMD field. All registers should be setup prior to writing a 1 to this bit position.*

Bit 5 (LAST_BYTE) default → 0

1 *= Software sets this bit to indicate that the next byte will be the last byte to be received for the block. This causes the ICH4 to send a NACK (instead of an ACK) after receiving the last byte.*

Bit 4:2 (SMB_CMD

011 *= Word Data: This command uses the transmit slave address, command, DATA0 and DATA1 registers. Bit 0 of the slave address register determines if this is a read or write command. If it is a read, after the command completes, the DATA0 and DATA1 registers will contain the read data.*

Bit 1 (KILL) default → 0

0 *= Normal SMBus Host controller functionality.*

Bit 0 (INTERN) default → 0

0 *= Disable.*

Host Command Register ----------03h

This 8-bit field is transmitted by the host controller in the command field of the SMBus protocol during the execution of any command.

Transmit Slave Address ---------- 04h

Bit 7:1 (ADDRESS) R/W →*7-bit address of the targeted slave.*

Bit 0 →R/W. *Direction of the host transfer.*

0 *→ Write /* ***1****→ Read*

Data 0 Register Address ----------05h

DATA0/COUNT *— R/W. This field contains the eight bit data sent in the DATA0 field of the SMBus protocol. For block write commands, this register reflects the number of bytes to transfer. This register should be programmed to a value between 1 and 32 for block counts. A count of 0 or a count above 32 will result in unpredictable behavior. The host controller does not check or log illegal block counts.*

Data 1 Register Address ----------06h

DATA1 *— R/W. This eight bit register is transmitted in the DATA1 field of the SMBus protocol during the execution of any command.*

PCA 9554 Device Address:

READ ----------4Eh

WRITE ----------4Fh

Command Description:

0x00 → Read byte – Input Port Register

0x01 → Read/Write byte – Output Port Register

0x02 → Read/Write byte – Polarity Inversion Register

0x04 → Read/Write byte – Configuration Register

Sample Code for reading SMB Registers in Assembly Language:

```
;=====================================================================
     NEWIODELAY Macro
                out  0ebh,al
                ENDM

;=====================================================================
                .model small
                .486p
                .stack 256
                .data
;=====================================================================
;                  Data Area
;=====================================================================
      SMBus_Port              EQU   500h
      PCA9554_ID              EQU  4eh  ;Device address
      Input_Reg               EQU  00h
      Output_Reg              EQU  01h  ;If GPIOx as output, the value
setting is effective.
      Inversion_Reg           EQU  02h
      Configure_Reg           EQU  03h  ;assign GPIOx as Input or
output.



;=====================================================================
;                Main Program Start
;=====================================================================

;=====================================================================
 .code
                 org     100h
 .STARTup

                 pusha


;           ;1.Set GPIO 0,1,2,3,4,5,6,7 as output
                 mov     ch,PCA9554_ID
                 mov     cl,Configure_Reg
                 mov     al,00000000b
                 call    Ct_I2CWriteByte
;

;           ;2.Set GPIO 0,2,4,6 Output Low & 1,3,5,7 Output High
                 mov     ch,PCA9554_ID
                 mov     cl,Output_Reg
                 mov     al,10101010b
                 call    Ct_I2CWriteByte


                 popa
                 .exit
```

```
;[]============================================================[]
;Input      :     CL - register index
;                 CH - device ID
;Output :         AL - Value read
;[]============================================================[]
Ct_I2CReadByte    Proc  Near

            push  cx

            mov   dx,SMBus_Port +04h
            inc   ch
            mov   al,ch             ;ID cmd(read)
            out   dx,al
            NEWIODELAY

            call  CT_Chk_SMBus_Ready

            pop   ax
            mov   dl,03h
            out   dx,al             ;Index
            NEWIODELAY

            mov   dl,02h
            mov   al,48h
            out   dx,al             ;Read data
            NEWIODELAY

            mov   cx, 10h

      @@:
            newiodelay
            loop  short @B

            call  CT_Chk_SMBus_Ready

            mov   dl,05
            in    al,dx             ;Data0
            NEWIODELAY

            ret
Ct_I2CReadByte    Endp

;[]============================================================[]
;Input :    CL - register index
;           CH - device ID
;           AL - Value to write
;Output:    none
;[]============================================================[]
Ct_I2CWriteByte   Proc  Near

            push  ax
            push  cx

            mov   dx,SMBus_Port +04h
            mov   al,ch             ;ID cmd(Write)
            out   dx,al
            call  Delay
```

```
            call  CT_Chk_SMBus_Ready

            pop   ax
            mov   dl,03h
            out   dx,al             ;Index
            call  Delay

            pop   ax
            mov   dl,05
            out   dx,al             ;Data0
            call  Delay


            mov   dl,02h
            mov   al,48h
            out   dx,al             ;write data
            call  Delay

            mov   cx, 10h
      @@:
            newiodelay
            loop  short @B

            call  CT_Chk_SMBus_Ready

            ret
Ct_I2CWriteByte   Endp

;;=====================================================================
CT_Chk_SMBus_Ready      Proc  Near
            mov   dx,SMBus_Port + 0;status port
            clc
            mov   cx,0800h
Chk_SMB_OK:
            in    al,dx       ;get status
            NEWIODELAY
            out   dx,al       ;clear status
            NEWIODELAY

            test  al, 02H           ;termination of command ?
            jnz   short Clear_final

            and   al, NOT 40H ;mask INUSE bit
            or    al,al       ;status OK ?
            jz    short Clear_final

            test  al,04h            ;device error
            jnz   short SMBus_Err

            loop  short Chk_SMB_OK
      ;SMbus error due to timeout
SMBus_Err:

            stc
            ret
Clear_final:
```

```
            clc
            ret
CT_Chk_SMBus_Ready      Endp


;;=============================================================================
Delay proc  near
            push  cx
            mov   cx, 100
      @@:
            NEWIODELAY
            loop  short @B
            pop   cx
            ret
Delay        ENDP

;=============================================================================
;              Program  END
;=============================================================================

        END
```