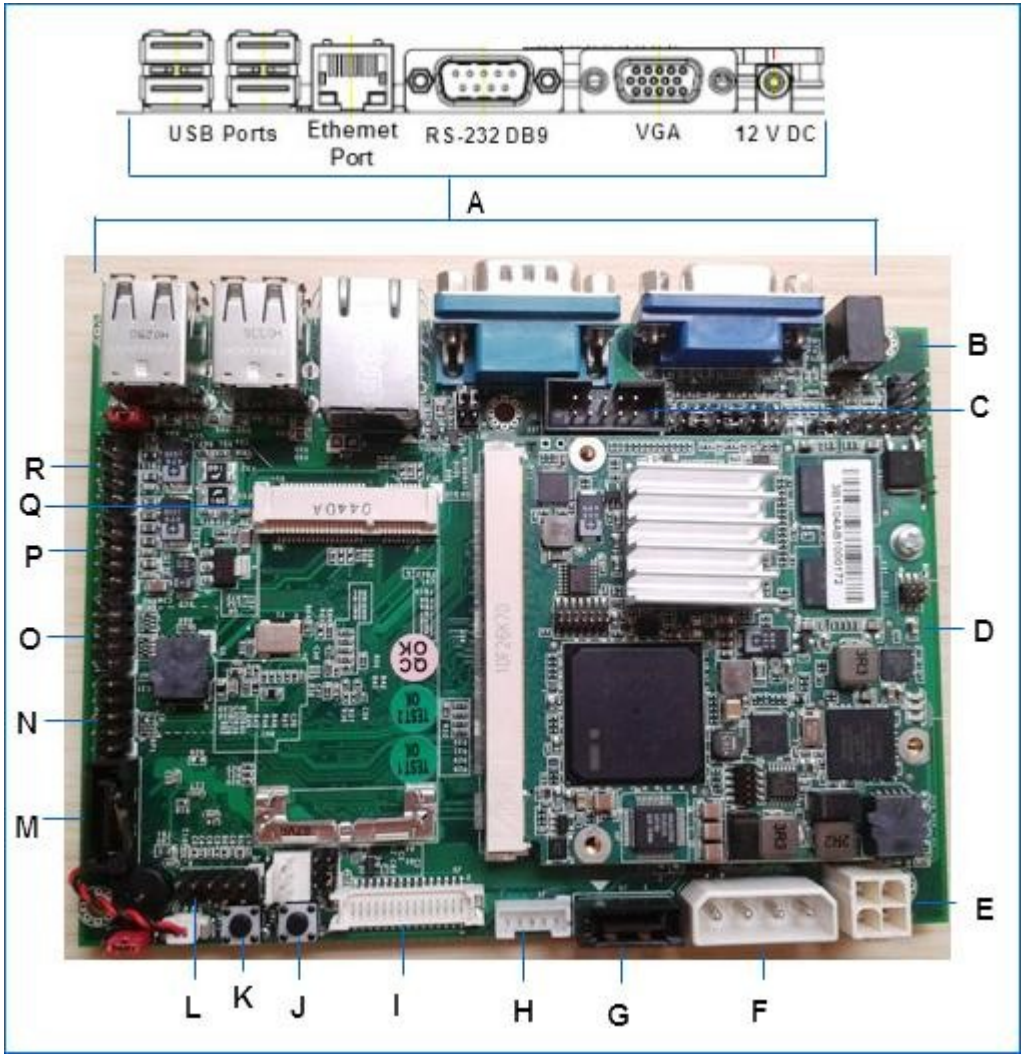
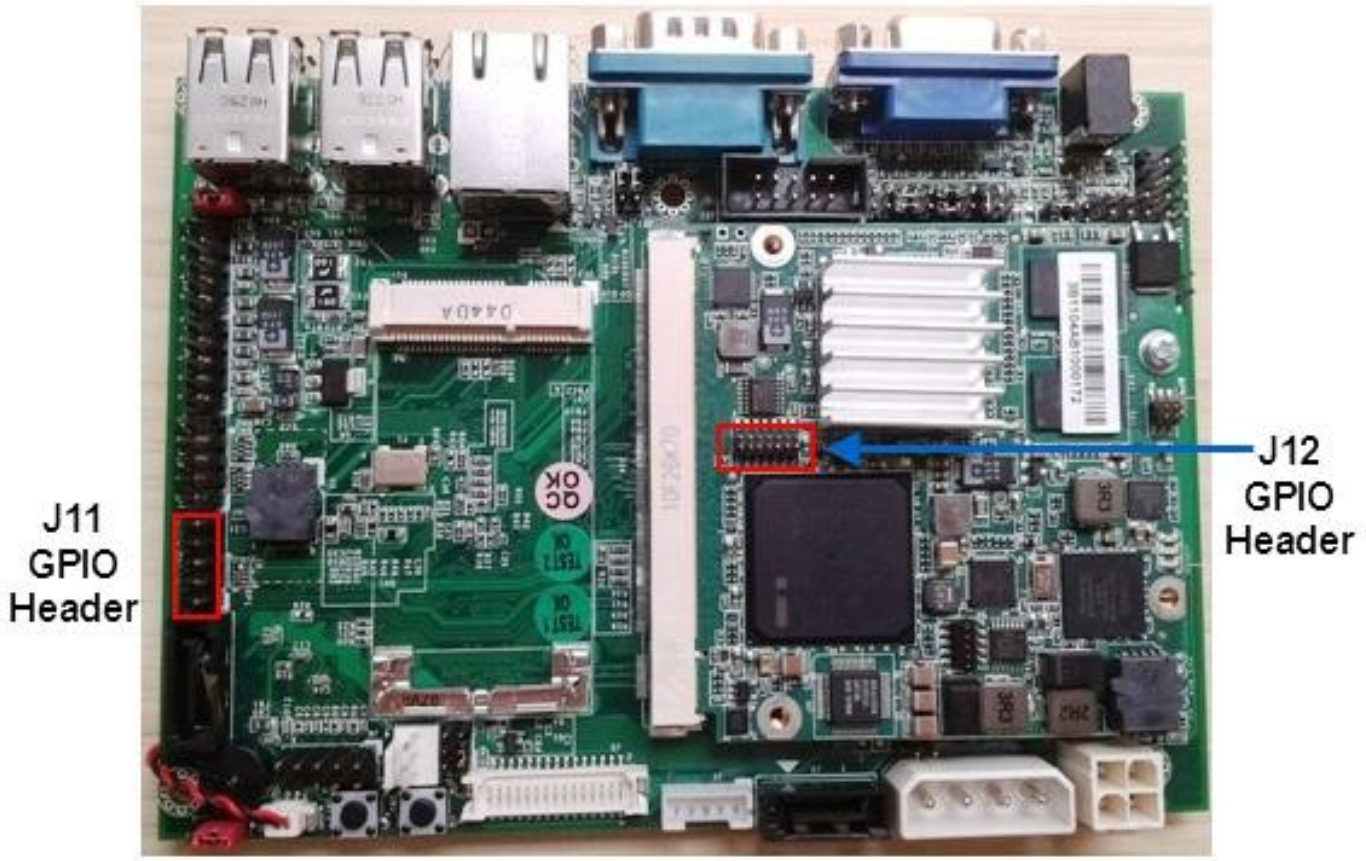
Programming GPIO using the Intel ATOM kits

System components



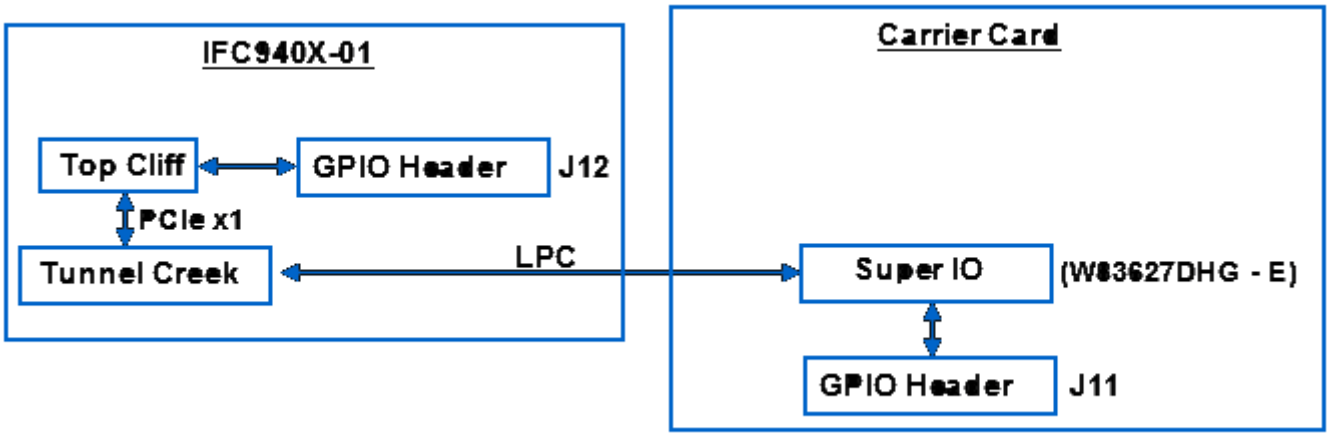
A Back panel connectors
B ECX Q7 carrier card
C RS232 header
D IFC940x-01 board
E Auxiliary power connector
F SATA power connector
G SATA DATA connector 0
H LVDS back light connector
I LVDS connector
J Restart switch
K Power switch
L Audio header
M SATA DATA connector 1
N GPIO header
O LPC debug port
P USB header 0
Q Mini PCIex1 connector
R USB header 1

System Components



GPIO Architecture

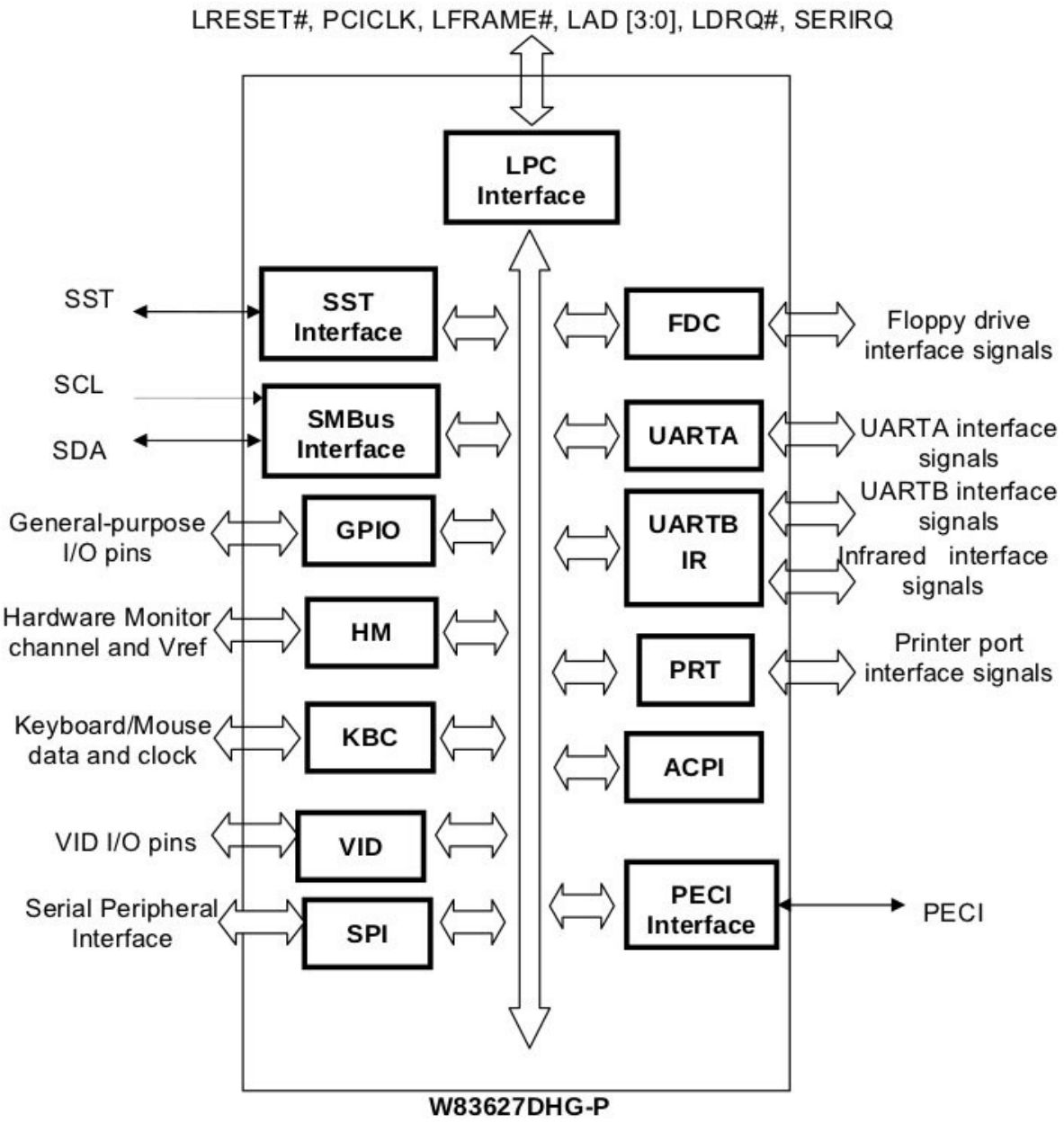
- General Purpose Input Output



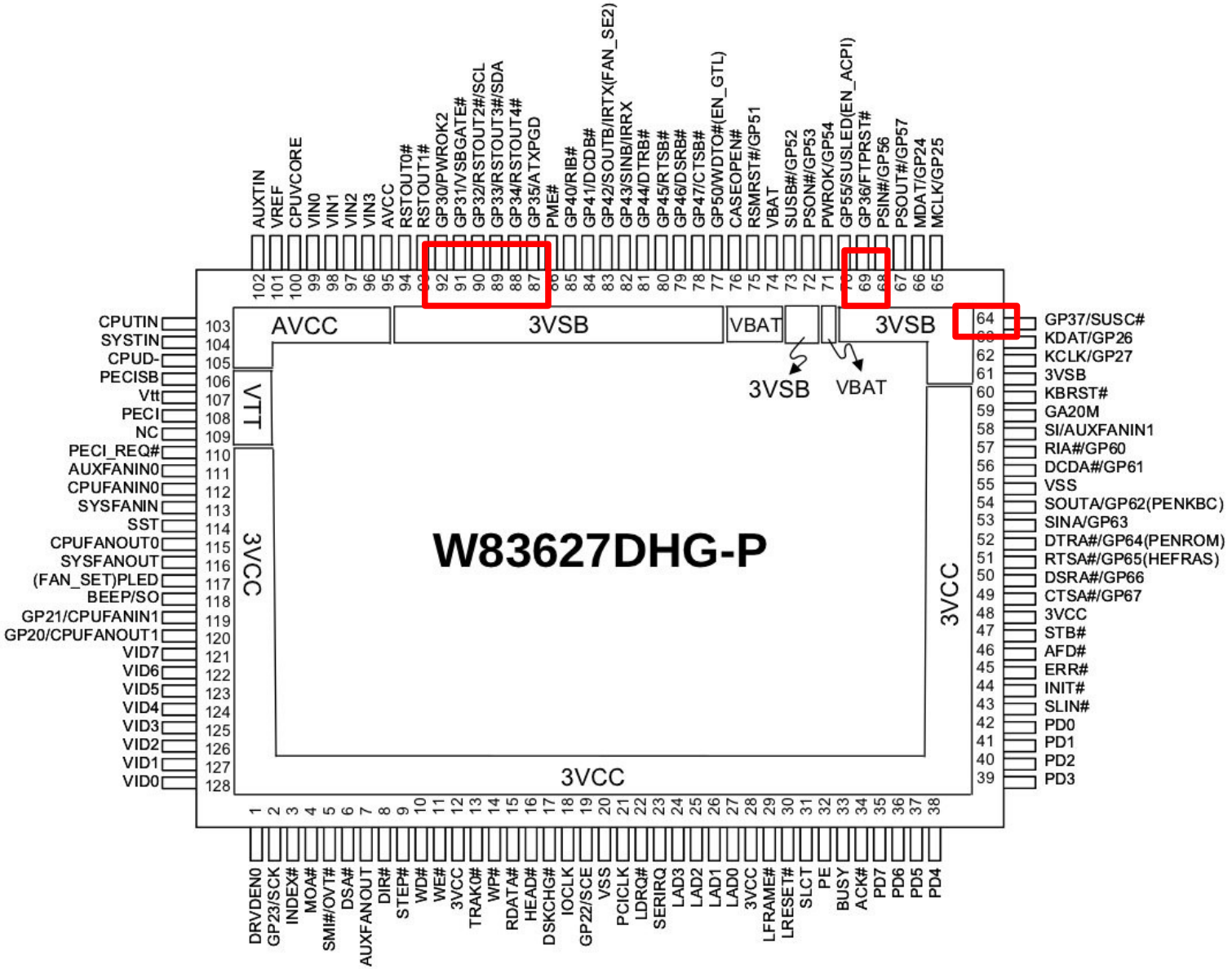
W83627DHG

- The W83627DHG is a member of Nuvoton's Super IO product line
- It is a Low Pin Count (LPC) interface
- It is economical than its ISA counterpart, and allows more efficient operation of software, BIOS and device drivers
- It also monitors several critical parameters like voltages, fan speeds, etc.
- It supports devices like disk drives, serial communication (UARTs), printer, serial interface, General Purpose I/O ports, etc.

Block diagram



Pin Layout for W83627DHG-P



Carrier Card (Super IO) GPIO header mapping

Signal Name	Device Location	Pin	Pull Up / Down
LPC GP34	Super IO – Pin # 88	2	2.7K to VCC
LPC GP35	Super IO – Pin # 87	4	2.7K to VCC
LPC GP36	Super IO – Pin # 69	6	2.7K to VCC
LPC GP37	Super IO – Pin # 64	8	2.7K to VCC
LPC GP30	Super IO – Pin # 92	1	2.7K to VCC
LPC GP31	Super IO – Pin # 91	3	2.7K to VCC
LPC GP32	Super IO – Pin # 90	5	2.7K to VCC
LPC GP33	Super IO – Pin # 89	7	2.7K to VCC

Pin details – J11 and J12

J11 GPIO header (HDR 2x5P 2.54mm)

on Carrier Card

Pin	Signal Name	Pin	Signal Name
1	LPC GP30	2	LPC GP34
3	LPC GP31	4	LPC GP35
5	LPC GP32	6	LPC GP36
7	LPC GP33	8	LPC GP37
9	GND	10	VCC=5V

J12 GPIO header (HDR 2x7P 1.27mm)

on CPU Module

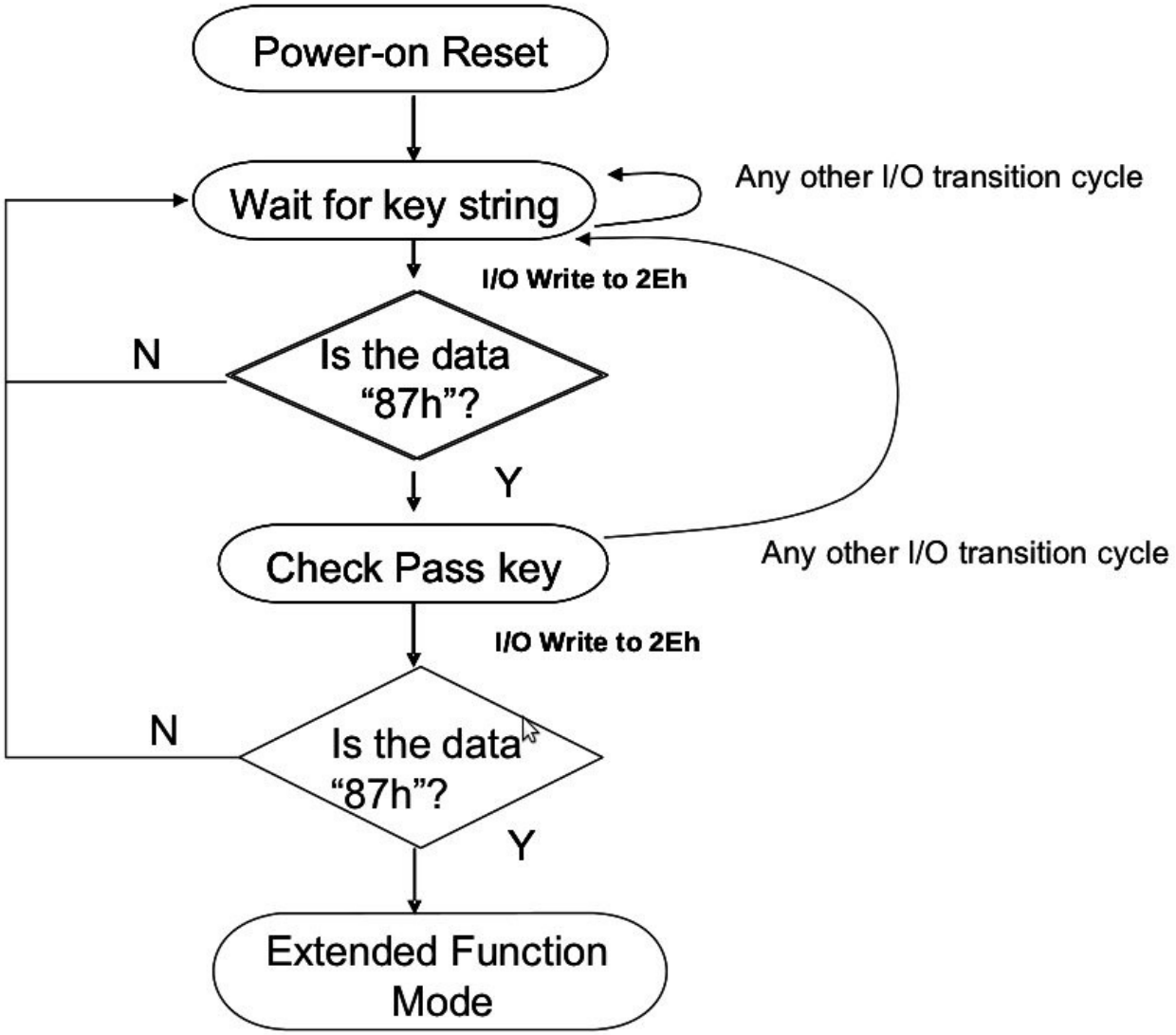
Pin	Signal Name	Pin	Signal Name
1	Reserved	2	V3P3=3.3V
3	IOH GPIO7	4	IOH GPIO 0
5	IOH GPIO8	6	IOH GPIO 1
7	IOH GPIO9	8	IOH GPIO 2
9	IOH GPIO10	10	IOH GPIO 3
11	IOH GPIO11	12	IOH GPIO 4
13	GND	14	IOH GPIO 5

Configuration register for Super IO

- The W83627DHG uses super I/O protocol to access configuration registers to set up different types of configurations. It has 12 logical devices
- Each device has its own configuration register
- The host can access these registers by writing an appropriate local device number into the logical device select register at CR07
- To program the W83627DHG configuration registers, follow the sequence:
  - Enter the Extended Function Mode
  - Configure the configuration registers
  - Exit the Extended Function Mode

Logical device	Function	I/O base address
0	FDC	100h ~ FF8h
1	Parallel Port	100h ~ FF8h
2	UART A	100h ~ FF8h
3	UART B	100h ~ FF8h
4	Reserved	
5	Keyboard Controller	100h ~ FFFh
6	Serial Peripheral Interface	100h ~ FF8h
7	GPIO 6	100h ~ FFFh
8	WDTO# & PLED	Reserved
9	GPIO 2, 3, 4, 5	Reserved
A	ACPI	Reserved
B	Hardware Monitor	100h ~ FFEh
C	PECI & SST	Reserved

Configuration Sequence



Configuration

- Enter the Extended Function Mode
  - To place the chip in extended function mode, two successive writes of 0x87H must be applied to the Extended Function Enable Registers (EFERs, i.e. 2EH or 4EH)
- Configure the configuration registers
  - First write the logical device number to the Extended Function Index Register (EFIR) (CR07) and then write the number of the desired logical device (0x09) in the extended function device register (EFDR).
  - Secondly, write the address of the desired configuration register within the logical device to the EFIR and then write (or read) the desired configuration register through the EFDR
- Exit the Extended Function Mode
  - Write 0xAA to EFER

Chip (Global) Control Registers

Index	R/W	Default value	Description
29h	R/W	00h	Multi-function Pin Selection
2Ah	R/W	00h	SPI Configuration
2Bh	Reserved		
2Ch	R/W	E2h	Multi-function Pin Selection
2Dh	R/W	21h	Multi-function Pin Selection

PORT3 GPIO pins

GPIO	Pin	Attribute	Multiple Function	Default Function	Power Plane
GP30	92	I/OD	N/A	GP30	VSB
GP31	91	I/OD	N/A	GP31	VSB
GP32	90	I/OD	SCL/RSTOUT2#	RSTOUT2#	VSB
GP33	89	I/OD	SDA/RSTOUT3#	RSTOUT3#	VSB
GP34	88	I/OD	RSTOUT4#	RSTOUT4#	VSB
GP35	87	I/OD	N/A	GP35	VSB
GP36	69	I/OD	N/A	GP36	VSB
GP37	64	I/OD	N/A	GP37	VSB

- GP 30, 31, 35, 36 = Pure GPIO and not multi function
- GP 32, 33, 34 = multi-function pins (shared by SMBus and Reset Out Buffer). So need to set them in GPIO mode by programming the configuration register CR2C

CR 2Ah							
7	6	5	4	3	2	1	0
Serial Peripheral Interface configuration						Pin 89, 90 function select =1 set by CR2C bits [6:5] =0 SDA, SCL	For GPIO port-2

CR 2Ch							
7	6	5	4	3	2	1	0
RW	RW	RW	RO	RW	RW	RW	R/W
0=GP34 1= PCI RSTOUT 4# Pin 88	0=GP33 1= PCI RSTOUT 3# Pin 89	0=GP32 1= PCI RSTOUT 2# Pin 90	Particular function, 0= disabled 1=enabled	EN_GTL configure bit VID input voltage 0= TTL 1=GTL	EN_PWRDN 0 = thermal shutdown disabled 1 = thermal shutdown enabed	To sets pins in reserved state, IRRX, IRTX, UART and GPIO4 modes	

CR 30h							
7	6	5	4	3	2	1	0
Reserved				R/W	R/W	R/W	R/W
				GPIO5	GPIO4	GPIO3	GPIO2
				0 = inactive 1 = active	0 = inactive 1 = active	0 = inactive 1 = active	0 = inactive 1 = active

Configuration registers

- Global control register = CR 07H is used for selecting logical device
  - Bits 0-7: read/write: identifies the logical device number
  - We need to select device number 9. Write 09h to the register
- Multi-function pin selection register = CR 2CH
  - We set bits 5,6,7 = 0, i.e. In GPIO mode. Other bits are kept un-altered
- PORT3 related configuration registers are:
  - CR2A[1] (R/W)
  - CR2C[5,6,7] (R/W)
  - Logical Device 9, CR30[1] (R/W)
  - Logical Device 9, CRF0 (input/outupt selection register, R/W)
    - Corresponding pin serves are 0=output, 1=input
  - Logical Device 9, CRF1 (Data register)
    - When a pin in PORT3 serves as input, the corresponding bit in this register will reflect its status. Writes to this bit will have no effect.
  - Logical Device 9, CRF2 (Inversion register)
  - Logical Device 9, CRE7(Status register)
  - Logical Device 9, CRFE (input detected type register)

Configure GPIO

```
unsigned char  IO_PORT_BASE = 0x2E;
unsigned char  DATA_PORT = IO_PORT_BASE + 1;

// Enter W83627HF Config
  outb(0x87,IO_PORT_BASE);
  outb(0x87,IO_PORT_BASE);

// Select GPIO/PCI mode by setting bit-1 of CR2A to '1'
  outb(0x2A,IO_PORT_BASE);
// in this register write the value of logical device number = 09h in our case
  outb(0x02,DATA_PORT);

// Set Multi-function Pins to GPIO, by configuring CR 2Ch
  outb(0x2C,IO_PORT_BASE);
// set bits 5,6,7=0 for GPIO....other bits un-altered
  outb((inb(DATA_PORT) & 0x1F),DATA_PORT);

// Locate logical device 9, by writing the number of device in the
// global control register CR07
  outb(0x07,IO_PORT_BASE);
// in this register write the value of logical device number = 09h in our case
  outb(0x09,DATA_PORT);

// from the selected logical device activate only GPIO-3 by configuring CR 30h
  outb(0x30,IO_PORT_BASE);
// write word 02H in the data to be sent to control register
  outb(0x02,DATA_PORT);
```

Configure GPIO

```
unsigned char  IO_PORT_BASE = 0x2E;
unsigned char  DATA_PORT = IO_PORT_BASE + 1;

// Enter W83627HF Config
  outb(0x87,IO_PORT_BASE);
  outb(0x87,IO_PORT_BASE);

// Select GPIO/PCI mode by setting bit-1 of CR2A to '1'
  outb(0x2A,IO_PORT_BASE);
// in this register write the value of logical device number = 09h in our case
  outb(0x02,DATA_PORT);

// Set Multi-function Pins to GPIO, by configuring CR 2Ch
  outb(0x2C,IO_PORT_BASE);
// set bits 5,6,7=0 for GPIO....other bits un-altered
  outb((inb(DATA_PORT) & 0x1F),DATA_PORT);

// Locate logical device 9, by writing the number of device in the
// global control register CR07
  outb(0x07,IO_PORT_BASE);
// in this register write
  outb(0x09,DATA_PORT);

// from the selected logical device activate only GPIO-3
  outb(0x30,IO_PORT_BASE);
// write word 02H in the data to be sent to control register
  outb(0x02,DATA_PORT);
```

Header files

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <fcntl.h>
#include <sys/io.h>
```

```
Void main()
{

 //permission to access the port
  if(ioperm(IO_PORT_BASE, 3, 1))
  {
     perror("ioperm");
     exit(1);
  }
  ...
}
```

Read GPIO

```
unsigned char  IO_PORT_BASE = 0x2E;
unsigned char  DATA_PORT = IO_PORT_BASE + 1;

// After configuration is complete
// we can read/write value '1' or '0' to the respective GPIO pins
// for GPIO-3 the input/output selection register is CR F0h
  outb(0xF0,IO_PORT_BASE);
  data=inb(DATA_PORT);

// read and print for every pin whether it is programmed as input or output
// GPIO-3 Data register is CR F1h

// Use the configuration in CR F0, to identify if pin is input or output
// and for input pins read the input value and display
  outb(0xF1,IO_PORT_BASE);
  data=inb(DATA_PORT);

// display input value on each pin

// Finally, Exit W83627HF Config by writing 0xAA in the base register
  outb(0xAA,IO_PORT_BASE);
```

Write GPIO

```
unsigned char  IO_PORT_BASE = 0x2E;
unsigned char  DATA_PORT = IO_PORT_BASE + 1;

// After configuration is complete
// we can read/write value '1' or '0' to the respective GPIO pins
// for GPIO-3 the input/output selection register is CR F0h
  outb(0xF0,IO_PORT_BASE);
  data=inb(DATA_PORT);

// modify 'data' and configure the pins to be either input or output
// as per the user preference
  outb(data,DATA_PORT);

// Read GPIO-3 Data register is CR F1h
  outb(0xF1,IO_PORT_BASE);
// read existing value
  data=inb(DATA_PORT);
// modify each bit to write a '0' or '1' on respective pins
// as per the pin configuration as either input or as output
  outb(data,DATA_PORT);

// Finally, Exit W83627HF Config by writing 0xAA in the base register
  outb(0xAA,IO_PORT_BASE);
```