LH7A400-10 GPIO Usage

White Paper 264

Mike Aanenson

Logic Product Development

Published: July, 2004

Abstract

This document outlines GPIO usage for Logic Product Development's LH7A400-10 Card Engine, PN 80000126.

REVISION HISTORY

REV	EDITOR	DESCRIPTION	APPROVAL	DATE
A	Mike Aanenson	Release	HAR	10/22/04

1 Introduction

This white-paper contains information to enable a LH7A400-10 user to:

1) Determine available GPIO signals from the LH7A400-10 Card Engine for the specific product application.

2) Find specific details on each GPIO signal that will allow the user to set up the signal as a GPIO signal.

1.1 Scope

This document is not meant to replace the Sharp LH7A400 User Guide, the Logic PD Hardware Spec, or the Logic PD I/O Controller Spec. Each of these documents should be used in collaboration with this white-paper.

This white paper provides <u>*example*</u> software code. This white paper does not serve as a programming guide with the exact software code to be used.

1.2 Important Notes

a) All references to "LH7A400" are to the Sharp processor, not the entire Logic Product Development card engine.

b) All references to "LH7A400-10" or "Card Engine" are references to the entire Logic Product Development Card Engine as a product.

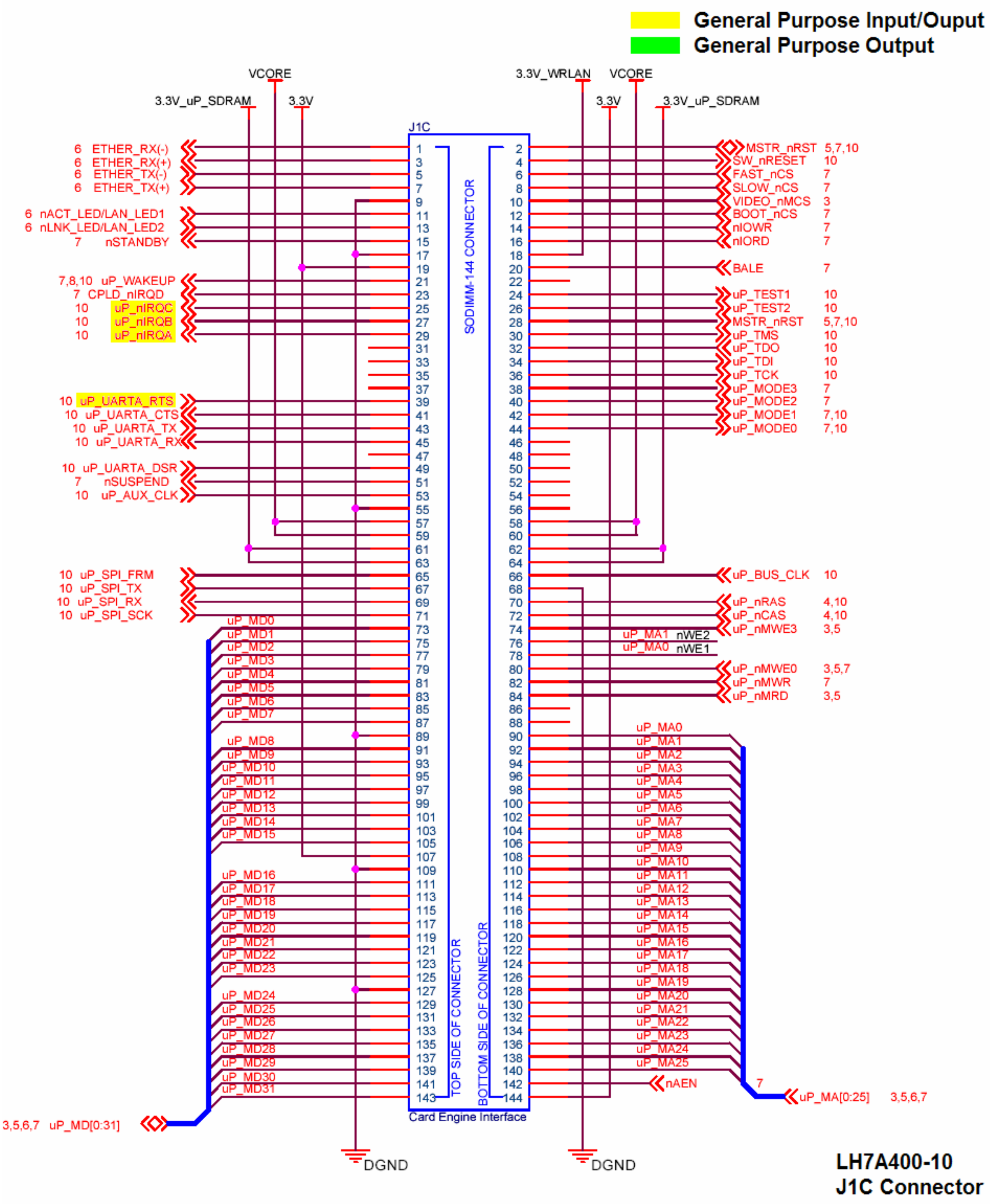
c) This document is updated to work with LH7A400-10 Rev B and greater card engines.

2 Overview

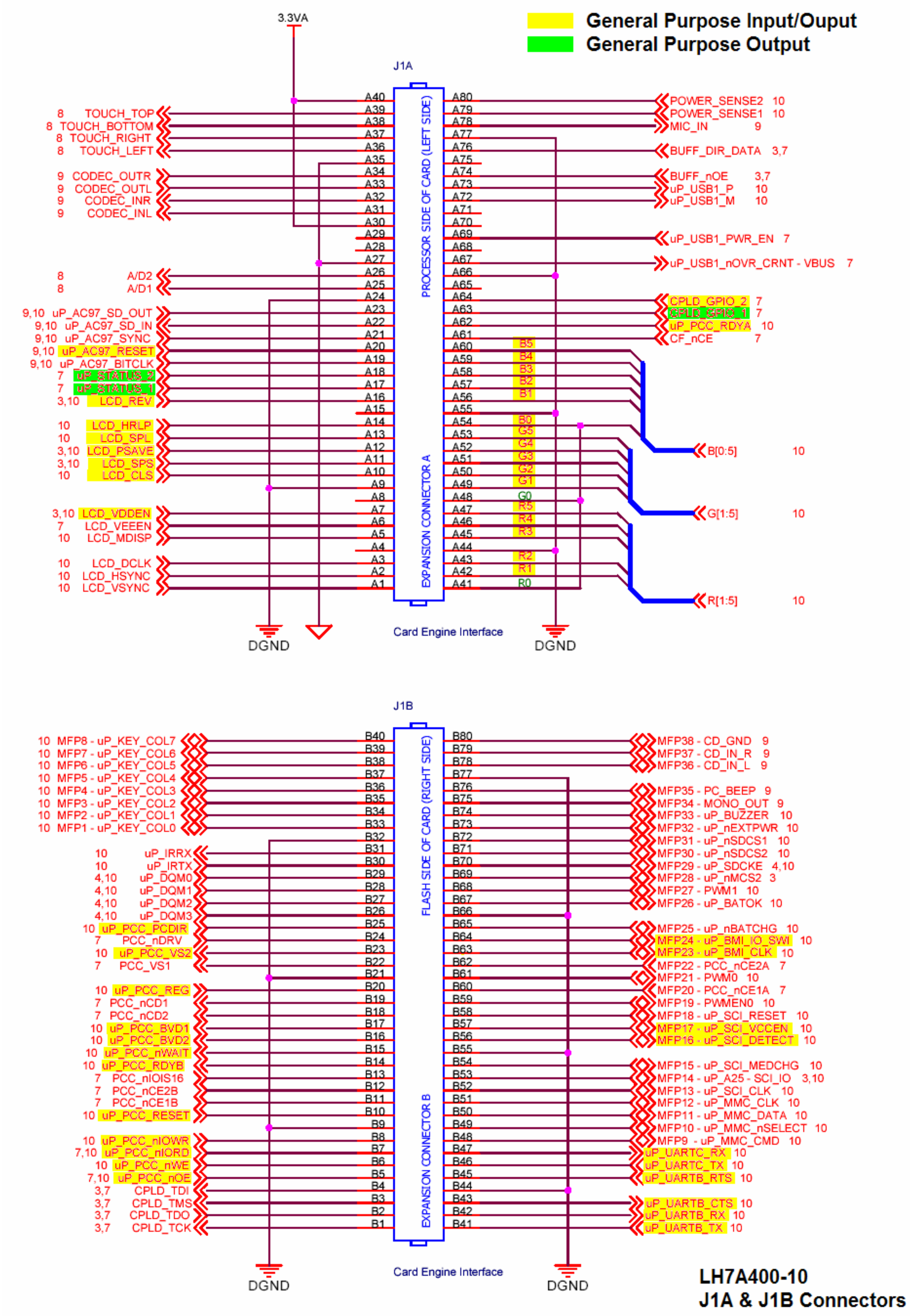
The LH7A400-10 card engine has many signals that are available to be used as GPIO, although many of these are multiplexed with a peripheral function. Through simple tables, this white-paper details each signal that is available as a GPIO. The information in these tables can be used to determine if the signal is being used as a peripheral function and if not, what needs to be done to set up the signal as a GPIO.

3 Available LH7A400-10 GPIO Signals Diagram

3.1 LH7A400-10 J1C Connector's Available GPIO



3.2 LH7A400-10 J1A and J1B Connector’s Available GPIO



4 LH7A400-10 GPIO Signals

4.1 GPIO Tables Header Information

This table presents table features and definitions.

Pin	LH7A400-10 Card Engine Pin Number
LH7A400-10 Signal Name	LH7A400-10 Card Engine Signal Name
LH7A400-10 Use	LH7A400-10's Primary Peripheral Use
Primary Description	LH7A400-10's Primary Peripheral Use Description
GPIO Configuration	LH7A400's GPIO Name
LH7A400 GPIO Description	LH7A400's GPIO Description
LH7A400 HW Reset	Reset state
LH7A400-10 LoLo Init Use	LogicLoader's initialization state

4.2 CPLD GPIO

CPLD GPIO

Pin	*LH7A400-10 Signal Name*	*LH7A400-10 Use*	*Primary Description*	*GPIO Configuration*	*LH7A400 GPIO Description*	*LH7A400 HW Reset*	*LH7A400-10 LoLo Init Use*
J1A.64	CPLD_GPIO_2	GPIO	General Purpose Input/Output	CPLD Data/Dir GPIO Register bit 0	CPLD General Purpose Input / Output	GPIO (o)	GPIO (o)
J1A.63	CPLD_GPIO_1	GPO	General Purpose Output	CPLD Extended GPIO Register bit 0	CPLD General Purpose Output	GPIO (o)	Status LED
J1A.18	uP_STATUS_2	LED Status	Status LED 2	CPLD Extended GPIO Register bit 1	CPLD General Purpose Output	GPIO (o)	Status LED
J1A.17	uP_STATUS_1	LED Status	Status LED 1	CPLD Extended GPIO Register bit 2	CPLD General Purpose Output	GPIO (o)	Status LED

Usage: The CPLD_GPIO_2 pin is a general purpose input/output that is controlled via a data register and a direction register in the CPLD. This signal is by default a GPIO set up as an output with a ‘low’ signal on reset.

The CPLD_GPIO_1 signal is a general purpose output that is controlled via the Extended GPIO Register in the CPLD. On reset, this pin is set to a ‘high’ level. On the LH7A400-10 SDK, the signal is connected to a LED. Logic’s software uses this pin as a LED status signal.

The uP_STATUS_2 and uP_STATUS_1 signals are used by Logic’s software as status LEDs. These pins are controlled via the Extended GPIO Register in the CPLD. On reset, these pins are set to ‘high’ levels. Both are connected to LEDs on the LH7A400-10 SDK.

Use of these CPLD signals is detailed in the *LH7A400-10 I/O Controller Spec*.

4.3 Interrupts as GPIO

Interrupt / GPIO							
Pin	*LH7A400-10 Signal Name*	*LH7A400-10 Use*	*Primary Description*	*GPIO Configuration*	*LH7A400 GPIO Description*	*LH7A400 HW Reset*	*LH7A400-10 LoLo Init Use*
J1C.25	uP_nIRQC	INT2	Interrupt 2 Input	PF2	Port F bit 2 I/O	GPIO (i)	GPIO (i)
J1C.27	uP_nIRQB	INT1	Interrupt 1 Input	PF1	Port F bit 1 I/O	GPIO (i)	GPIO (i)
J1C.29	uP_nIRQA	INT0	Interrupt 0 Input	PF0	Port F bit 0 I/O	GPIO (i)	GPIO (i)

Usage: These pins can be used as GPIO individually, meaning that you do not need to choose to use either the entire group as GPIO or not. The LH7A400 on restart will set these up as GPIO and Logic Product Development's software (LogicLoader, Bootloader, etc.) will leave these as GPIO on initialization.

The process and details of using these signals as GPIO is detailed in the LH7A400 User Guide from Sharp.

4.4 UART Signals as GPIO

UARTA / GPIO							
Pin	*LH7A400-10 Signal Name*	*LH7A400-10 Use*	*Primary Description*	*GPIO Configuration*	*LH7A400 GPIO Description*	*LH7A400 HW Reset*	*LH7A400-10 LoLo Init Use*
J1C.39	uP_UARTA_RTS**	UARTA RTS	UARTA RTS	PA6	Port A bit 6 I/O	GPIO (i)	GPIO (i)

** Important Note: The uP_UARTA_RTS signal is a GPIO pin, although Logic has reserved it as a UARTA signal for use in some software instances. If Logic's software is not used, this pin may be configured as a GPIO pin (rather than a UARTA signal) even if UARTA is being used.

Usage: This UARTA pin is a LH7A400 GPIO pin that Logic PD has reserved for use as the RTS signal for UARTA. Some of Logic PD's software uses it in this fashion, although LogicLoader does not. If the software running does not use this signal as part of the UARTA interface, it can be used as a GPIO signal. The LH7A400 processor on reset will set it up as a GPIO.

The process and details of using these signals as GPIO is detailed in the LH7A400 User Guide from Sharp.

UARTB / GPIO							
Pin	*LH7A400-10 Signal Name*	*LH7A400-10 Use*	*Primary Description*	*GPIO Configuration*	*LH7A400 GPIO Description*	*LH7A400 HW Reset*	*LH7A400-10 LoLo Init Use*
J1B.41	uP_UARTB_TX	UARTTX3	UART B TX Output Only	PB1	Port B bit 1 I/O	GPIO (i)	UARTTX3

J1B.42	uP_UARTB_RX	UARTRX3	UART B RX Input Only	PB2	Port B bit 2 I/O	GPIO (i)	UARTRX3
J1B.43	uP_UARTB_CTS	UARTCTS3	UART B RX Clear To Send	PB3	Port B bit 3 I/O	GPIO (i)	UARTCTS3
J1B.45	uP_UARTB_RTS**	UARTRTS3	UART B RequestTo Send	PA7	Port A bit 7 I/O	GPIO (i)	GPIO (i)

** Important Note: The uP_UARTB_RTS signal is a GPIO pin, although it has been reserved as a UARTB signal. If the software does not use this signal as part of the UARTB interface, it can be used as a GPIO signal. The LH7A400 processor on reset will set it up as a GPIO.

Usage: The three UARTB signals TX, RX, and CTS can be configured as a group as being either GPIO or UARTB signals. These can not be different individually. On reset, the LH7A400 processor sets these to be GPIO. Logic PD's software initializes to be UART signals.

The uP_UARTB_RTS signal is a GPIO pin, although it has been reserved as a UARTB signal. If the software does not use this signal as part of the UARTB interface, it can be used as a GPIO signal.

The process and details of using these signals as GPIO is detailed in the LH7A400 User Guide from Sharp.

UARTC / GPIO

Pin	*LH7A400-10 Signal Name*	*LH7A400-10 Use*	*Primary Description*	*GPIO Configuration*	*LH7A400 GPIO Description*	*LH7A400 HW Reset*	*LH7A400-10 LoLo Init Use*
J1B.46	UP_UARTC_TX	UARTTX1	UART C TX Output Only	PC0	Port C bit 0 I/O	GPIO (o)	GPIO (o)
J1B.46	UP_UARTC_RX	UARTRX1	UART C RX Input Only	PB0	Port B bit 0 I/O	GPIO (i)	GPIO (i)

Usage: The two UARTC signals TX and RX can be configured as a group as being either GPIO or UARTC signals. These can not be different individually.

The process and details of using these signals as GPIO is detailed in the LH7A400 User Guide from Sharp.

4.5 LCD Signals as GPIO

LCD / GPIO							
Pin	*LH7A400-10 Signal Name*	*LH7A400-10 Use*	*Primary Description*	*GPIO Configuration*	*LH7A400 GPIO Description*	*LH7A400 HW Reset*	*LH7A400-10 LoLo Init Use*
J1A.7	LCD_ VDDEN	LCDVDDEN	LCDVDDEN (Digital supply enable)	PC2	Port C bit 2 I/O	GPIO (o)	GPIO (o)
J1A.10	LCD_CLS	LCDCLS	ADTFT/HRTFT LCD CLS Signal Output (Gate Driver Clock)	PC5	Port C bit 5 I/O	GPIO (o)	GPIO (o)
J1A.11	LCD_SPS	LCDFP	ADTFT/HRTFT LCD SPS (row reset)	PC4	Port C bit 4 I/O	GPIO (o)	GPIO (o)
J1A.12	LCD_PSAVE	LCDPS	ADTFT/HRTFT LCD PS (power save)	PC1	Port C bit 1 I/O	GPIO (o)	GPIO (o)
J1A.13	LCD_SPL	LCDSPL	ADTFT/HRTFT LCD SPL Signal Output (line start pulse left)	PC7	Port C bit 7 I/O	GPIO (o)	GPIO (o)
J1A.14	LCD_HRLP	LCDHRLP	ADTFT/HRTFT LCD HRLP Signal Output (horizontal sync pulse)	PC6	Port C bit 6 I/O	GPIO (o)	GPIO (o)
J1A.16	LCD_REV	LCDREV	ADTFT/HRTFT LCD REV Signal Output (AC bias)	PC3	Port C bit 3 I/O	GPIO (o)	GPIO (o)
J1A.47	R5	LCDVD4	LCD Data 4 I/O	PE0	Port E bit 0 I/O	GPIO (i)	LCDVD4
J1A.49	G1	LCDVD5	LCD Data 5 I/O	PE1	Port E bit 1 I/O	GPIO (i)	LCDVD5
J1A.50	G2	LCDVD6	LCD Data 6 I/O	PE2	Port E bit 2 I/O	GPIO (i)	LCDVD6
J1A.51	G3	LCDVD7	LCD Data 7 I/O	PE3	Port E bit 3 I/O	GPIO (i)	LCDVD7
J1A.52	G4	LCDVD8	LCD Data 8 I/O	PD0	Port D bit 0 I/O	GPIO (o)	LCDVD8
J1A.53	G5	LCDVD9	LCD Data 9 I/O	PD1	Port D bit 1 I/O	GPIO (o)	LCDVD9
J1A.54, J1A.48, J1A.41	B0,R0,G0	LCDVD15	LCD Intensity (same signal on these pins)	PD7	Port D bit 7 I/O	GPIO (o)	LCDVD15
J1A.56	B1	LCDVD10	LCD Data 10 I/O	PD2	Port D bit 2 I/O	GPIO (o)	LCDVD10
J1A.57	B2	LCDVD11	LCD Data 11 I/O	PD3	Port D bit 3 I/O	GPIO (o)	LCDVD11
J1A.58	B3	LCDVD12	LCD Data 12 I/O	PD4	Port D bit 4 I/O	GPIO (o)	LCDVD12
J1A.59	B4	LCDVD13	LCD Data 13 I/O	PD5	Port D bit 5 I/O	GPIO (o)	LCDVD13
J1A.60	B5	LCDVD14	LCD Data 14 I/O	PD6	Port D bit 6 I/O	GPIO (o)	LCDVD14

Usage: These LCD signals can be configured as a group as being either GPIO or LCD signals. If the LCD interface is being used, none of these pins can be used as GPIO.

The process and details of using these signals as GPIO is detailed in the LH7A400 User Guide from Sharp.

4.6 AC97 Signals as GPIO

AC97 Audio / GPIO							
Pin	*LH7A400-10 Signal Name*	*LH7A400-10 Use*	*Primary Description*	*GPIO Configuration*	*LH7A400 GPIO Description*	*LH7A400 HW Reset*	*LH7A400-10 LoLo Init Use*
J1A.20	uP_AC97_RESET	AC97RESET	AC97 reset signal	PH6	Port H bit 6 I/O	AC97RESET	AC97RESET

Usage: This AC97 signal can be configured as being either a GPIO or AC97 signal, although if the AC97 interface is being used, it can not be used as a GPIO.

The process and details of using these signals as GPIO is detailed in the LH7A400 User Guide from Sharp.

4.7 PCMCIA Signals as GPIO

PCMCIA / GPIO							
Pin	*LH7A400-10 Signal Name*	*LH7A400-10 Use*	*Primary Description*	*GPIO Configuration*	*LH7A400 GPIO Description*	*LH7A400 HW Reset*	*LH7A400-10 LoLo Init Use*
J1A.62	uP_PCC_RDYA	PCRDYA	PC Card Ready Signal A	PH7	Port H bit 7 I/O	GPIO (i)	PCRDYA
J1B.5	uP_PCC_nOE	nPCOE	PC Card Output Enable	PG0	Port G bit 0 I/O	GPIO (o)	nPCOE
J1B.6	uP_PCC_nWE	nPCWE	PC Card Write Enable	PG1	Port G bit 1 I/O	GPIO (o)	nPCWE
J1B.7	uP_PCC_nIORD	nPCIOR	PC Card IO Write Output	PG2	Port G bit 2 I/O	GPIO (o)	nPCIOR
J1B.8	uP_PCC_nIOWR	nPCIOW	PC Card IO READ Output	PG3	Port G bit 3 I/O	GPIO (o)	nPCIOW
J1B.10	uP_PCC_RESET	PCRESET1	PC Card Reset 1 Output	PH0	Port H bit 0 I/O	GPIO (i)	PCRESET1
J1B.14	uP_PCC_RDYB	PCRDYB	PC Card Ready Signal B	PH6	Port H bit 6 I/O	GPIO (i)	PCRDYB
J1B.15	uP_PCC_nWAIT	nPCWAIT1	PC Card wait signal	PH4 / PH5	Port H bit 4 I/O Port H bit 5 I/O	GPIO (i)	nPCWAIT1

					(tied to both pins for use as dual PCMCIA/CF WAIT signals. If using as GPIO, use only Port H bit 4)		
J1B.16	uP_PCC_BVD2	PCBVD2	PC Card Battery Voltage Detect 2	PA5	Port A bit 5 I/O	GPIO (i)	GPIO (i)
J1B.17	uP_PCC_BVD1	PCBVD1	PC Card Battery Voltage Detect 1	PA4	Port A bit 4 I/O	GPIO (i)	GPIO (i)
J1B.20	uP_PCC_REG	PCREG	PC Card Reg	PG4	Port G bit 4 I/O	GPIO (o)	PCREG
J1B.23	uP_PCC_VS2	nPCSLOTE2	PC Card Voltage Sense 2 Input	PA3	Port A bit 3 I/O	GPIO (i)	GPIO (i)
J1B.25	uP_PCC_PCDIR	PCDIR	CF and PCMCIA Direction	PG7	Port G bit 7 I/O	GPIO (o)	GPIO (o)

Usage: These PCMCIA signals can be configured as a group as being either GPIO or PCMCIA signals. If the PCMCIA interface is being used, none of these pins can be used as GPIO. Please be aware that if the port G and H signals are to be used as GPIO, the GPACT bit of the GPIO Direction Register in the CPLD must be set to 1. If this is not done, the card engine may lock up.

The process and details of using these signals as GPIO is detailed in the LH7A400 User Guide from Sharp. The CPLD GPACT bit is detailed in the *LH7A400-10 I/O Controller Spec* from Logic PD.

4.8 Smart Card Signals as GPIO

Smart Card / GPIO							
Pin	*LH7A400-10 Signal Name*	*LH7A400-10 Use*	*Primary Description*	*GPIO Configuration*	*LH7A400 GPIO Description*	*LH7A400 HW Reset*	*LH7A400-10 LoLo Init Use*
J1B.56	MFP16 – uP_SCI_DETECT	SCDETECT	Smart Card detection	PF5 or INT5	Port F bit 5 I/O or Interrupt 5 Input	GPIO (i)	GPIO (i)
J1B.57	MFP17 – uP_SCI_VCCEN	SCVCCEN	Smart Card Supply Voltage Enable	PF4 or INT4	Port F bit 4 I/O or Interrupt 4 Input	GPIO (i)	GPIO (i)

Usage: These Smart Card signals can be configured as a group as being either GPIO or Smart Card signals. If the Smart Card interface is being used, none of these pins can be used as GPIO.

The process and details of using these signals as GPIO is detailed in the LH7A400 User Guide from Sharp.

4.9 Smart Battery Signals as GPIO

Smart Battery / GPIO							
Pin	*LH7A400-10 Signal Name*	*LH7A400-10 Use*	*Primary Description*	*GPIO Configuration*	*LH7A400 GPIO Description*	*LH7A400 HW Reset*	*LH7A400-10 LoLo Init Use*
J1B.63	MFP23 – uP_BMI_CLK	SMBCLK	Smart Battery Clock	PB7	Port B bit 7 I/O	GPIO (i)	GPIO (i)
J1B.64	MFP24 – uP_BMI_IO_SWI	SMBD	Smart Battery Data	PB6 or SWID	Port B bit 6 I/O or Single Wire Data	GPIO (i)	GPIO (i)

Usage: These Smart Battery signals can be configured as a group as being either GPIO or Smart Battery signals. If the Smart Battery interface is being used, none of these pins can be used as GPIO.

The process and details of using these signals as GPIO is detailed in the LH7A400 User Guide from Sharp.

5 Example Setup and Usage

5.1 GPIO Usage

In order to properly use a GPIO signal, follow the following steps:

1) Determine if the GPIO is multiplexed with another onboard peripheral.
    a. This document can be used in conjunction with the Sharp LH7A400 User Guide, Logic PD LH7A400 HW Spec and Logic PD I/O Controller spec.
2) Disable Peripheral Functionality
    a. The Sharp LH7A400 User Guide, Section “Pin and Signal Multiplexing”, explains for each peripheral how it is disabled.
    b. It is recommended to make sure the software disables the peripheral, even if it is known to be set this way be default.
3) Setup GPIO direction
    a. The Sharp LH7A400 User Guide, Section “GPIO and External Interrupts”, details each GPIO port’s direction and data registers.
    b. It is recommended to make sure the software sets up the desirable direction of the GPIO, even if it is known to be set this way by default.

5.2 C code example

The following example will show how to set the MFP23 – uP_BMI_CLK signal up as a GPIO signal and then set it to be an output with a ‘high’ value.

```
/* LH7A400's Register Address Defines – From LH7A400 User Guide */
#define PBDDR       (*(unsigned long volatile *)(0x80000E14))
#define PBDR        (*(unsigned long volatile *)(0x80000E04))
#define SWICR       (*(unsigned long volatile *)(0x80000F04))
#define SBICR       (*(unsigned long volatile *)(0x80000F44))

void example_gpio_setup(void)
{
    /*Disable Battery Monitor Interface Signals */
    /* Set SWICR bit 0 (SWIEN) to 0, disable Single Wire Interface */
    SWICR &= 0xFFFE;

    /* Set SBICR bit 0 (SBIEN) to 0, disable Smart Battery Interface */
    SBICR &= 0xFFFE;

    /*Set pin to be output via the port’s data direction register (PyDD)*/
    PBDDR |= 0x0080;   //port b bit 7 hi = output
    /*Set pin to be a ‘high’ level via the port’s data register (PyD) */
    PBDR |= 0x0080;    // port b bit 7 hi = ‘high’ value

    return;
}
```

5.3 Windows CE app/driver example

The following example will show how to set the MFP23 – uP_BMI_CLK signal up as a GPIO signal and then set it to be an output with a ‘high’ value. It doesn’t contain any error checking, just the basics.

```
/* LH7A400's Battery Monitor Register Defines */
#define BMI_REGS                                    (0x80000F00)
#define BMI_REGS_SIZE                               (0x5B)
#define BMI_SWICR_REG_OFFSET                        (0x04)
#define BMI_SBICR_REG_OFFSET                        (0x44)

/* LH7A400's GPIO Register Defines */
#define GPIO_REGS                                   (0x80000E00)
#define GPIO_REGS_SIZE                              (0x87)
#define GPIO_PBDD_REG_OFFSET                        (0x14)
#define GPIO_PBD_REG_OFFSET                         (0x04)

void example_gpio_setup_function(void)
{
    volatile BYTE *bmi_address_base  = NULL;        /* Points to Pin Function register.*/
    volatile BYTE *gpio_address_base = NULL;        /* Points to Pin Function register*/
    ULONG tmp;

    //Map in physical addresses for LH7A400 registers to be virtual addresses.
    phy_addr.LowPart  = BMI_Regs;
    phy_addr.HighPart = 0;
    bmi_address_base = MmMapIoSpace(phy_addr, BMI_REGS_SIZE, FALSE);

    phy_addr.LowPart  = GPIO_REGS;
    phy_addr.HighPart = 0;
    gpio_address_base = MmMapIoSpace(phy_addr, GPIO_REGS_SIZE, FALSE);

    //Disable Battery Monitor Interface Signals
```

```
        //Set SWICR bit 0 (SWIEN) to 0       //disable Single Wire Interface
        tmp = READ_REGISTER_ULONG(bmi_address_base + BMI_SWICR_REG_OFFSET)
        WRITE_REGISTER_ULONG((bmi_address_base+BMI_SWICR_REG_OFFSET), tmp & 0xFFFFFFFE);
        //Set SBICR bit 0 (SBIEN) to 0        //disable Smart Battery Interface
        tmp = READ_REGISTER_ULONG(bmi_address_base + BMI_SBICR_REG_OFFSET)
        WRITE_REGISTER_ULONG((bmi_address_base+BMI_SBICR_REG_OFFSET), tmp & 0xFFFFFFFE);


        //Set pin to be output via the port’s data direction register (PyDD)
        //PBDD = 0x80000E14 bit 7 set to 1 for output
        tmp = READ_REGISTER_ULONG(gpio_address_base + GPIO_PBDD_REG_OFFSET)
        WRITE_REGISTER_ULONG((gpio_address_base+GPIO_PBDD_REG_OFFSET), tmp | 0x00000080);

        //Set pin to be a ‘high’ level via the port’s data register (PyD)
        //PBD = 0x80000E04 bit 7 is set to 1 for a ‘high’ value.
        tmp = READ_REGISTER_ULONG(gpio_address_base + GPIO_PBD_REG_OFFSET)
        WRITE_REGISTER_ULONG((gpio_address_base+GPIO_PBDD_REG_OFFSET), tmp | 0x00000080);

        return;
}
```