Memory and Input/Output Systems in the S3CEV40 Board.

Contents

Chapter 1

System Memory

Figure 1.1 represents schematically the memory subsystem of the train-board used in the lab [1]. The operation of the memory system actually depends on three components in the laboratory board, (ARM7TDMI) processor, (S3C44BOX) system on chip on which it is integrated and the board on which the latter (S3CEV40) is placed. We will describe the main features of each of its components.

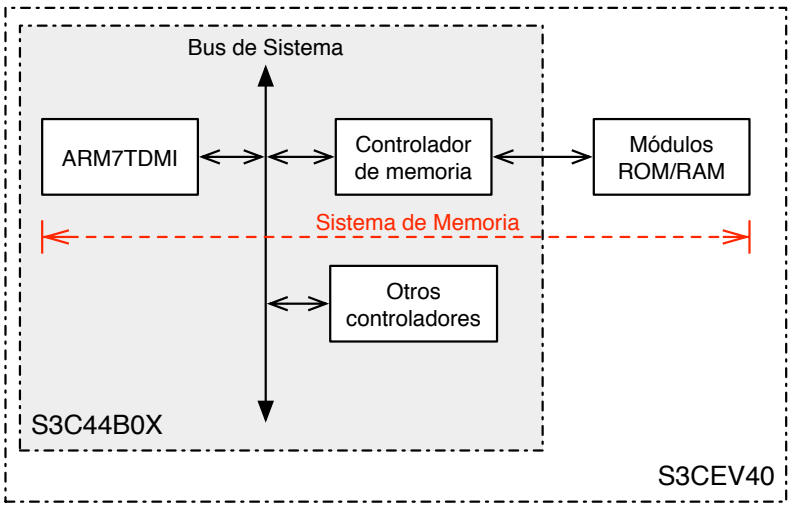

Figure 1.1: System memory laboratory board (Embest S3CEV40).

1.1 The ARM7TDMI

The ARM7TDI address bus is 32 bits width, it can potentially address a total space of 4GB of memory. It must also be noted that this address space is shared by the system I/O (*I/O located in memory*), threfore the addresses may concern both spaces of main memory as internal elements of the I/O controllers (local memory or registers). In both cases the

[1] Although the *SoC* S3C44B0X has a *Write Buffer* y una 8K *Cache*, on default both are disabled.

ARM7TDMI accesses in same way and is up to the programmer to know which device is associated with each address. This assignment of address ranges to devices often referred as *system memory map*.

1.2 The memory controller

The memory controller is responsible to interface between (ROM or RAM) external memory modules and the system bus. It is easy to guess that the *system memory map* is mainly determined by the characteristics and configuration of this element of the memory system.

The behavior of the memory controller could be summarized as follows. When the processor gets an address on the bus:

- If the address is within the range of the driver, then it is responsible for generating the necessary signals to access to the corresponding memory module.
- If, however, the address is outside its range, the controller is inhibited, since may other I/O controller be responsible on such access (read/write of a register or local memory).

If the memory access fails (eg accessing to a bank for which there is NOT allocated any memory module), the controller is responsible on detect the error and generate the corresponding exception *Abort*.

1.2.1 The S3C44B0X memory controller

In laboratory board (Embest S3CEV40), both the ARM7TDMI processor and the memory controller, and other I/O controllers, are integrated into a single chip, the Samsung S3C44B0X *System on Chip*.

The S3C44B0X memory controller reduces the address space into 256MB, and divides it into 8 separate fragments, referreing them as *banks*, as shown in Figure 1.2. Each bank can be assigned to a different external memory chip (module), although only the last two banks admit any type of memory (SROM, SRAM or SDRAM). When accessing to an address within the range of one of these banks, besides necessary signals for access to the module to which it is associated, the memory controller activates one NGCs signal[2]. These signals are used externally to select/activate the corresponding chip.

The top of the first bank, corresponding to the range `0x01C00000-0x01FFFFFF`, is reserved for I/O ports of the integrated controllers within the S3C44B0X. As mentioned previously, when the processor performs an access within this range, the memory controller is inhibited (ie does not generate any signal).

1.3 Memory Modules

The board used in the laboratory (Embest S3CEV40) has two different types of modules:

[2]By convention, signals whose name starts with "n" are activated in low logic level.

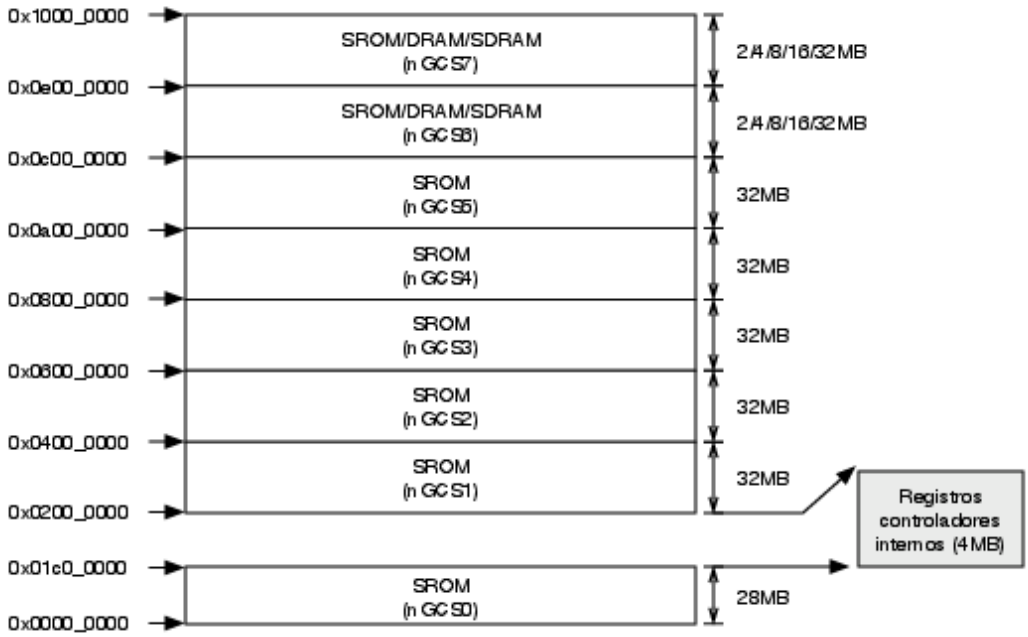

Figure 1.2: Mapa de memoria del S3C44B0X (SROM se refiere tanto a ROM como a SRAM).

- 1Mx16bits Flash ROM memory, located in the bank 0 of the S3C44B0X, and whose address range is `[0x00000000-0x001FFFFF]`.
- 4Mx16bits SDRAM memory located on the bank 6 S3C44B0X and whose address range is `[0x0C000000-0x0C7FFFFF]`.
-

As we can see, the 256MB space allowed by the memory controller is reduced to 10MB, 2MB Flash ROM and 8MB SDRAM memories.

Chapter 2

Input/Output System

Figure 2.1 represents schematically the input/output system in the board used in the laboratory. Its main components are the ARM7TDMI processor, interrupt controller, I/O controllers – both internal and external to the Samsung S3C44B0X –, and selection logic, which uses the signals generated by the memory controller to select the external chip to use. We will briefly describe the main features of each of its components.

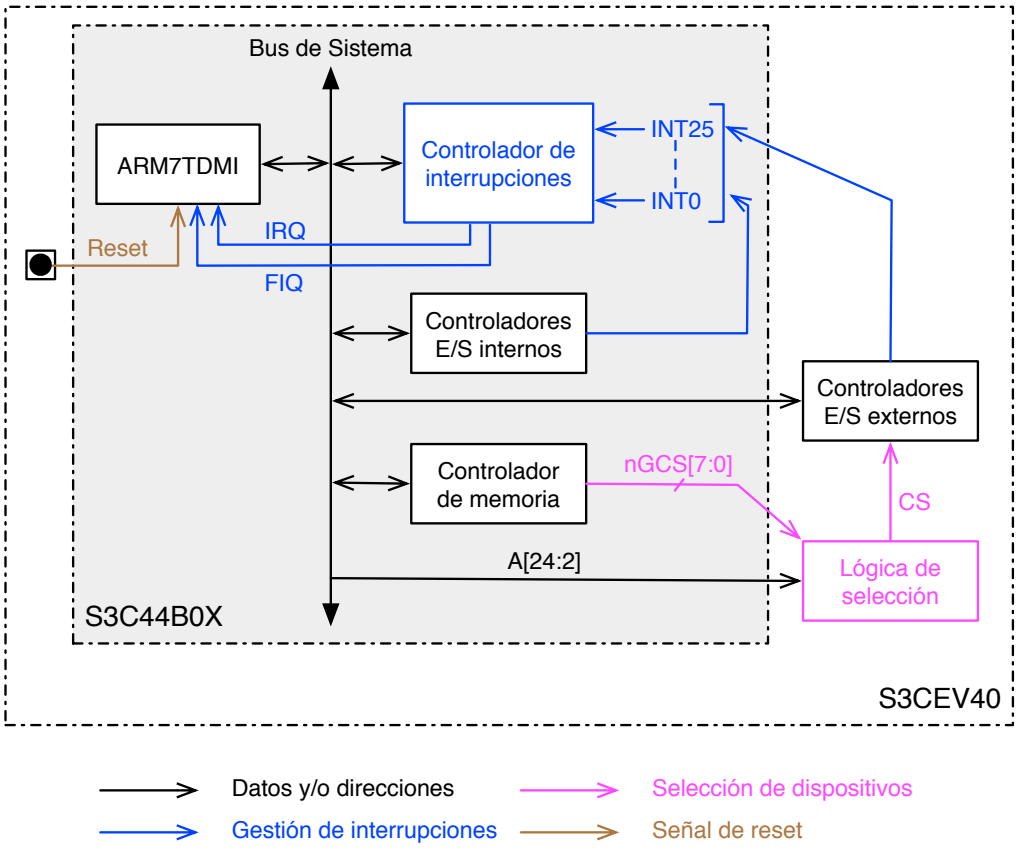

Figure 2.1: Sistema de E/S de la placa de laboratorio.

2.1 I/O Controllers and Devices (GPIO)

Usually (there are only few exceptions), the physical I/O is managed through specific I/O controllers. Those act as an interface between the device, the system bus, and the interrupt controller. To communicate with the device, the processor reads and writes on the internal registers of these controllers, which we commonly refer them as I/O ports.

Selection of devices/controllers

As previously mentioned, the ARM7TDMI processor uses I/O located in memory and therefore is not able to distinguish whether an address corresponds to an I/O or a memory address. The memory controller plays an essential role in differentiate between these two ranges: a range of addresses assigned to the registers to the corresponding internal S3C44B0X drivers (I/o, memory, interrupts, ...) and the remaining 256MB of address space which addresses memory and is responsible for managing, creating necessary signals for communication with the module, and activating the NGCs signal for the bank (nGCS0-nGCS7).

In addition to internal controllers, there are external I/O system controllers. The I/O ports corresponding to these external controllers to S3C44B0X are located outside the range corresponding to internal addresses and drivers, requiring additional logic for its selection (see Figure 2.1). Obviously, there can be assigned I/O ports to banks with memory modules. This selection logic is responsible for activating the signal *Chip Select* (CS) for the device to be accessed based on the value of the nGCS0-7 signals and address bits in the system bus.

Chapter 3

I/O pins controller(GPIO)

General Purpose I/O pin driver (GPIO) keeps the control of 71 multifunctional pins, ie, each of these pins can be used for more than one different function. The GPIO allows setting the functionality for each of these pins.

The 71 pins are divided into 7 sets, referred as ports, which are:

- 2 sets of I/O 9 bits (E and F ports)
- 2 sets of I/O 8 bits (D and G ports)
- 1 set of I/O 16 bits (C port)
- 1 set of I/O 11 bits (B port)
- 1 set of I/O 10 bits (A port)

Each port is managed by 2-4 registers. The actual number depends one each port. When we *Reset* the system, these registers loads a safe value to prevent any possible damage on the system or hardware connected to these pins. In the lab exercises, we will use only some of the I/O pins of port B and G. Table 3.1 contains a list of their control and data registers, which we describe in detail below. Studens must consult the Manual S3C44B0X [um-] for a complete description.

Table 3.1: Registers of the B and G ports

Register	**Address**	**R/W**	**Reset value**
PCONB	0x01D20008	R/W	0x7FF
PDATB	0x01D2000C	R/W	Undef
PCONG	0x01D20040	R/W	0x00
PDATG	0x01D20044	R/W	Undef
PUPG	0x01D20048	R/W	0x00
EXTINT	0x01D20050	R/W	0x00000000
EXTINTPND	0x01D20054	R/W	0x00

3.1 Port B

The port B has 11 bits, which support two features: these pins can used as output pins (to write a value in them) or can connect to some signals generated by the memory controller (*ie* these signals go outside of S3C44B0X using these pins). The port configuration is performed through two GPIO registers:

- *PCONB*, control register that selects the functionality of each of the pins. Its description is in Table 3.2.
- *PDATB*, data register that allows to write the bits to take from the port (only useful for those pins configured as output pins).

Table 3.2: Configuration of the pins of the port B.

PCONB	Bit	Description
PB10	[10]	0 = Output 1 = nGCS5
PB9	[9]	0 = Output 1 = nGCS4
PB8	[8]	0 = Output 1 = nGCS3
PB7	[7]	0 = Output 1 = nGCS2
PB6	[6]	0 = Output 1 = nGCS1
PB5	[5]	0 = Output 1 = nWBE3/nBE3/DQM3
PB4	[4]	0 = Output 1 = nWBE2/nBE2/DQM2
PB3	[3]	0 = Output 1 = nSRAS/nCAS3
PB2	[2]	0 = Output 1 = nSCAS/nCAS2
PB1	[1]	0 = Output 1 = SCLK
PB0	[0]	0 = Output 1 = SCKE

In the practical exercises, we use two pins in the port B configured as output pins(pins 9 and 10, as discussed below) to turn on or off two LEDs connected to them.

3.2 Port G

Port G has eight bits that can be configured in four different ways. The port configuration is performed using three GPIO registers:

- *PCONG*,control register for selecting the functionality of each pin, as described in Table 3.3.
- *PDATG*, data regioter that allows writing or reading from the port G.
- *PUPG*,configuration register that activates (bit to '0') or not (bit to '1') a *pull-up* resistor[1] for each pin.

[1] A *pull-up* resistor puts the entrance to one when it is disconnected, avoiding an undefined value. Similarly, a *pull-down* resistor puts the entrance to zero when the input is disconnected.

Table 3.3: Configuration of the pins of the port G.

PCONG	Bits	Description	
PG7	15:14	00 = Input	01 = Output
		10 = IISLRCK	11 = EINT7
PG6	13:12	00 = Input	01 = Output
		10 = IISDO	11 = EINT6
PG5	11:10	00 = Input	01 = Output
		10 = IISDI	11 = EINT5
PG4	9:8	00 = Input	01 = Output
		10 = IISCLK	11 = EINT4
PG3	7:6	00 = Input	01 = Output
		10 = nRTS0	11 = EINT3
PG2	5:4	00 = Input	01 = Output
		10 = nCTS0	11 = EINT2
PG1	3:2	00 = Input	01 = Output
		10 = VD5	11 = EINT1
PG0	1:0	00 = Input	01 = Output
		10 = VD4	11 = EINT0

As seen in the table, we can use the bits of port G as input or output pins or connect them to some internal system signals. Alternatively, we can configure the port pins to activate the interrupt request `EINT*` lines. This allows some external hardware, connected to these pins, can generate an interrupt using these lines. In this way, this port supports various options for triggering the interrupt. The interrupt can be generated when a voltage level is detected on the pin or when a particular edge is detected. This is configured by a fourth register of GPIO, *EXTINT* whose use is described in Table 3.4.

The GPIO controller also allows to decompose the line EINT4/5/6/7 of the interrupt controller into four lines, which operation will be explained later. The EINT4/5/6/7 line from the interrupt handler/controller is activated when any of these four lines generates an interrupt (actually performing an OR). The *EXTINTPND* register from the port G allows to know which of the four lines has generated the interrupt as described in Table 3.5.

It should be emphasized that to clear the interrupt request by one of these interrupt lines, the service routine must clear the corresponding bit in this register (writing a '1', not a '0') before clearing the bit INTPND of the interrupt controller.

Later we will explain how to use the G port to read the buttons or to generate an interrupt when it detects that one of the buttons has been pressed.

3.3 LEDs and buttons

The S3CEV40 board has two buttons and two LEDs connected to S3C44BOX system as shown in Figure 3.1.

The LEDs are connected to pins 9 and 10 on port B. As shown in Figure 3.1, its anode connected to the power supply (Vdd) to illuminate these LEDs is necessary to configure

Table 3.4: Registro *EXTINT*.

EXTINT	Bit	Description
EINT7	[30:28]	Establece el método de señalización de EINT7. 000 = Interrupción por nivel bajo 001 = Interrupción por nivel alto 01x = Disparado por flanco de bajada 10x = Disparado por flanco de subida 11x = Disparado por ambos flancos
EINT6	[26:24]	Establece el método de señalización de EINT6. 000 = Interrupción por nivel bajo 001 = Interrupción por nivel alto 01x = Disparado por flanco de bajada 10x = Disparado por flanco de subida 11x = Disparado por ambos flancos
EINT5	[22:20]	Establece el método de señalización de EINT5. 000 = Interrupción por nivel bajo 001 = Interrupción por nivel alto 01x = Disparado por flanco de bajada 10x = Disparado por flanco de subida 11x = Disparado por ambos flancos
EINT4	[18:16]	Establece el método de señalización de EINT4. 000 = Interrupción por nivel bajo 001 = Interrupción por nivel alto 01x = Disparado por flanco de bajada 10x = Disparado por flanco de subida 11x = Disparado por ambos flancos
EINT3	[14:12]	Establece el método de señalización de EINT3. 000 = Interrupción por nivel bajo 001 = Interrupción por nivel alto 01x = Disparado por flanco de bajada 10x = Disparado por flanco de subida 11x = Disparado por ambos flancos
EINT2	[10:8]	Establece el método de señalización de EINT2. 000 = Interrupción por nivel bajo 001 = Interrupción por nivel alto 01x = Disparado por flanco de bajada 10x = Disparado por flanco de subida 11x = Disparado por ambos flancos
EINT1	[6:4]	Establece el método de señalización de EINT1. 000 = Interrupción por nivel bajo 001 = Interrupción por nivel alto 01x = Disparado por flanco de bajada 10x = Disparado por flanco de subida 11x = Disparado por ambos flancos
EINT0	[2:0]	Establece el método de señalización de EINT0. 000 = Interrupción por nivel bajo 001 = Interrupción por nivel alto 01x = Disparado por flanco de bajada 10x = Disparado por flanco de subida 11x = Disparado por ambos flancos

the pins 9 and 10 as output pins (see Table 3.2) and write a '0' on them using the register *PDATB*.

The two switches are connected to pins 6 and 7 of port G. If these pins are set as input pins, then we will read a '0' in the corresponding register bit in the PDATG when the button is pressed. To ensure that the pins take the logic value '1' (Vdd) when the button is not pressed, it is needed to activate the pull-up resistors on these pins through the PUPG register. However, a clean drop in voltage Vdd to 0 will not occur when a button be pressed, but the voltage will bounces (in a range between Vdd and 0), until finally stabilizes at 0. This can make system detect multiple times pressed although where has been pressed only one time. To filter these bounces, system should wait a while when is detected a pulse before detect the next one. The waiting time can be adjusted empirically, but usually 100ms delay is enough. This waiting time can be implemented with a Delay function which basically consists of two nested loops, with just an empty body in the inner loop.

Table 3.5: Purpose of the register bits *EXTINTPND*.

EXTINTPND	Bit	Description
EXTINTPND3	[3]	When EINT7 is activated, EXINTPND3 se pone a 1, y INTPND[21] se pone a 1.
EXTINTPND2	[2]	When EINT6 is activated, EXINTPND2 se pone a 1, y INTPND[21] se pone a 1.
EXTINTPND1	[1]	When EINT5 is activated, EXINTPND1 se pone a 1, y INTPND[21] se pone a 1.
EXTINTPND0	[0]	When EINT4 is activated, EXINTPND0 se pone a 1, y INTPND[21] se pone a 1.

Alternatively, the pins 6 and 7 of G port connected to the buttons can also be connected to the EXINT6 and EXINT7 lines of interrupt controller by setting the appropriate value at the *PCONG* register (see Table 3.3). To generate interrupts using these buttons, it is also necessary to enable the *pull-up* resistors of these pins through the *PUPG* register (to maintain constant the input value when the pin is in the air) otherwise, the behavior would be unpredictable. It is also necessary to set the interrupt type (high / low, rising/falling/both edge) through the *EXTINT* register

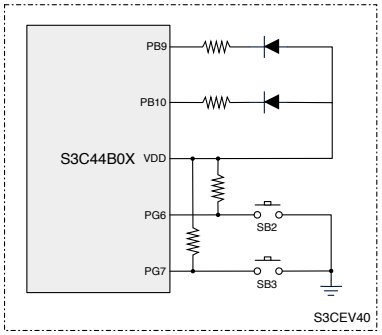

Figure 3.1: Esquema de conexión de pulsadores y LEDs de la placa Embest S3CEV40.

Chapter 4

8 Segment Display

The board S3CEV40 also has a display of 8 segments, consisting of 7 LEDs to form any hexadecimal digit and an additional LED for the decimal point. This display is common anode type, it means that inputs must put to zero to turn on the LEDs. Figure 4.1 shows the connection of the display to the external selection logic. The LEDs on the display are connected to the data bus of the processor through a tristate latch (74LS573 chip). The selection of this device is carried out through CS6 signal, generated by a 3-8 decoder(chip 74LV138), which is enabled with nGCS1 signal according to the scheme of Figure 4.1. This selection logic is common to all devices on the Bank-1. Table 4.1 shows the address range and selection signal assigned to each of them. As can be shown, the latch connected to the 8-segment display is activated with the CS6 signal each time the bus access to an address in the range 0x0214_0000-0x0217_FFFF. While the LE (Latch Enable) input remains activates the latch will pass the byte defined by the 8 least significant bits of the data bus. When this signal turns off the latch holds the last value. However, bank-1 has a width of 8 bits. It means that when writting a word size in the memory controller, it will send separately each of the 4 bytes that compose the word. Since the processor is configured as little-endian, bytes be written in order of their weight (0,1,2 and 3), thus the latch will kepp the most significant byte, instead of the less significant. For this reason, it is very important to always writes only on byte size to the 8-segment display, by using the STRB instruction.

Table 4.1: Rango asignado a cada uno de los dispositivos ubicados en el Banco-1.

Device	**CS**	**Address**
USB	CS1	0x0200_0000 - 0x0203_FFFF
Nand Flash	CS2	0x0204_0000 - 0x0207_FFFF
IDE (IOR/W)	CS3	0x0208_0000 - 0x020B_FFFF
IDE (KEY)	CS4	0x020C_0000 - 0x020F_FFFF
IDE (PDIAG)	CS5	0x0210_0000 - 0x0213_FFFF
8-SEG	CS6	0x0214_0000 - 0x0217_FFFF
ETHERNET	CS7	0x0218_0000 - 0x021B_FFFF
LCD	CS8	0x021C_0000 - 0x021F_FFFF

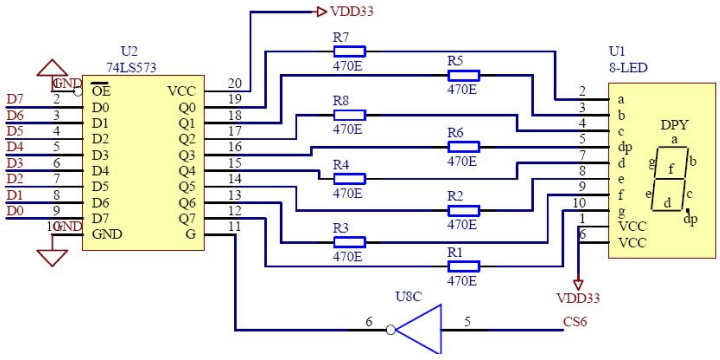

Figure 4.1: Wiring diagram of the 8-segment display on the S3CEV40 board. Bits on positions A[20:18] are address bits while the bits on positions D[7: 0] are data bits in S3C44B0X system bus. keep in mind that the nGCS1 memory controller signal is activated at low level when addressing a reference in the Bank-1.

Chapter 5

Timers

The S3C44B0X has 6 timers of 16-bits, connected as shown in Figure 5.1. Each of them can be configured to operate interrupt or DMA. Timers 0, 1, 2, 3 and 4 can generate a square wave of the frequency set on the timer, Pulse Width Modulation (PWM), are the TOUT* signals in the figure 5.1. These signals can be removed through a pin for transmitting a signal for pulse width modulation, motor control, etc. This functionality is not available in the timer 5, which can only be used for internal timing and has no output pin. Registers used for handling timers are shown in Table 5.1.

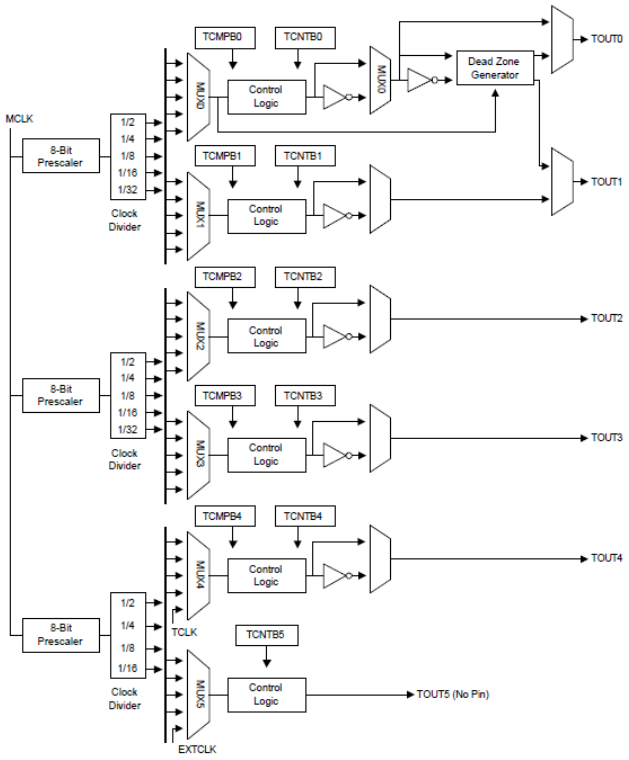

Figure 5.1: Wiring diagram of timers in the S3C44B0X.

Operation of the timers

Timers are down counters that can be set to a certain value. Once configured and initialized, and decremented each cycle of internal clock. When one of them reaches 0, generates an interrupt that can be used to perform some tasks periodically.

As shown in Figure 5.1, each pair of timers (0-1, 2-3, 4-5) shares a pre-scaling module and a frequency divider. The divisor of the first 4 timers has five distinct divided signals: $1/2, 1/4, 1/8, 1/16$ and $1/32$. Timers 4 and 5 have a divider with four divided clock signals: $1/2, 1/4, 1/8$ and $1/16$, and a further input TCLK/EXTCLK, used to perform the features of a signal different from the clock signal (for example to count pulses external to the system, obtained from a pin). As shown, the dividers are fed with the output of pre-scaling modules , then the couting frequency is constructed with a dividing process in two stages, as will be detailed later.

Each timer (except number 5) has a pair of associated registers, TCNTn and TCMPn. The counting register(TCNTn) is initialized to a certain value and is decremented each

Table 5.1: Registers for the use of timers

Register	Address	R/W	Reset Value
TCFG0	0x01D50000	R/W	0x00000000
TCFG1	0x01D50004	R/W	0x00000000
TCON	0x01D50008	R/W	0x00000000
TCNTB0	0x01D5000C	R/W	0x00000000
TCMPB0	0x01D50010	R/W	0x00000000
TCNTO0	0x01D50014	R	0x00000000
TCNTB1	0x01D50018	R/W	0x00000000
TCMPB1	0x01D5001C	R/W	0x00000000
TCNTO1	0x01D50020	R	0x00000000
TCNTB2	0x01D50024	R/W	0x00000000
TCMPB2	0x01D50028	R/W	0x00000000
TCNTO2	0x01D5002C	R	0x00000000
TCNTB3	0x01D50030	R/W	0x00000000
TCMPB3	0x01D50034	R/W	0x00000000
TCNTO3	0x01D50038	R	0x00000000
TCNTB4	0x01D5003C	R/W	0x00000000
TCMPB4	0x01D50040	R/W	0x00000000
TCNTO4	0x01D50044	R	0x00000000
TCNTB5	0x01D50048	R/W	0x00000000
TCNTO5	0x01D5004C	R	0x00000000

cycle while the timer be active. The compare register (TCMPn) is initialized to another value, it is compared with the counting register. The result of this comparisson is used to control the output signal. This signal has a value of 0 at the beginning of each count, but signal is set to 1 when the counter reaches the value of comparisson register. In this way we can control exactly the pulse width (PWM), ie the number of cycles must be at 0 and 1.

The timers can operate in two modes: *auto-reload* and *one-shot*. Mode *auto-reload* when the timer reaches zero, the initial value is automatically reloaded and begins a new countdown. This serves to generate very accurately periodic interrupts. In the *one-shot*, when the counter reaches zero the count stops.

Figure 5.2 represents the time diagram of an example of normal operation of the timers. It uses a double buffer, it allows user to modify the timer reload value while it is counting, without altering or stopping the counter. Normally, the timer is initialized by writing a value in the count register (TCNTn) and activating the *manual update bit*, as discussed in the following section. While the counter is running, it is modified the value of the register of the associated buffer (TCNTBn) and the count value will not be change. This new value is used to recharge TCNTn when the count reaches zero (if mode is *auto-reload*). The same happens on the compare register (TCMPn), it will be reloaded from the buffer register (TCMPBn).

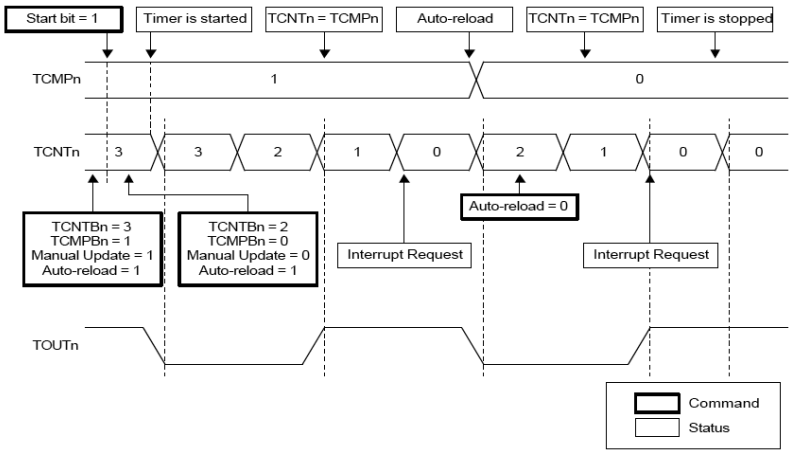

Figure 5.2: Diagrama temporal del funcionamiento de los temporizadores del S3C44B0X.

The current count value can be read from the observation register (TCNTOn). The value When reading from TCNTBn the value obtained is not the current value of the count register, but the reload value.

Initializing Timers

The value of the register TCNTBn is copied couter register TCNTn ONLY when TCNTn reaches zero, the timer can not be initialized by simply writting in TCNTBn. To forward the value wrote in TCNTBn to TCNTn must be activated the *manual update bit*. The timer initialization sequence must be:

1. Write the initial values of count and comparisson in TCNTBn and TCMPBn, respectively.
2. Turn on *manual update bit* of the timer.
3. Turn on the *start bit* at same time that be disabled the *manual update bit*.

Setting of Timers

Effective Clock signal for timers is:

$$F = MCLK/\left((\text{valor de pre-escalado} + 1)(\text{valor del divisor})\right)$$

where MCLK is the internal clock signal, pre-scaled value is in the $0 - 255$ range and value of the divisor: $2, 4, 8, 16, 32$. The value of pre-scaling for the timers are configured in the registry TCFG0 described in Table 5.2.

The value of the divider is configured with the register TCFG1 and assigment of a timer to the request DMA channel. The description of the register appears in Table 5.3.

The timer control (start, stop, update, auto-reload mode, ...) is done through the TCON register. Its functionality is described in Table 5.4.

Table 5.2: Register *TCFG0*.

Function	**Bits**	**Description**
Longitud de la zona muerta	[31:24]	Estos 8 bits determinan la zona muerta. La unidad de tiempo de la zona muerta es la misma que la del temporizador 0.
Pre-escalado 2	[23:16]	Estos ocho bits determinan el factor de pre-escalado de los temporizadores 4 y 5.
Pre-escalado 1	[15:8]	Estos ocho bits determinan el factor de pre-escalado de los temporizadores 2 y 3.
Pre-escalado 0	[7:0]	Estos ocho bits determinan el factor de pre-escalado de los temporizadores 0 y 1.

Finally, buffer registers must be used to manage the timers, both count and comparison registers, and also the observation register.

For further information on timers must consult the S3c44B0X Manual [um-].

Table 5.3: Register *TCFG1*.

Function	**Bits**	**Description**
modo DMA	[27:24]	Selecciona el canal de DMA 0000 = No seleccionado 0001 = Timer0 0010 = Timer1 0011 = Timer2 0100 = Timer3 0101 = Timer4 0110 = Timer5 0111 = Reserved
MUX 5	[23:20]	Selecciona el MUX para el Timer5. 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 01xx = EXTCLK
MUX 4	[19:16]	Selecciona el MUX para el Timer4. 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 01xx = TCLK
MUX 3	[15:12]	Selecciona el MUX para el Timer3. 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 01xx = 1/32
MUX 2	[11:8]	Selecciona el MUX para el Timer2. 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 01xx = 1/32
MUX 1	[7:4]	Selecciona el MUX para el Timer1. 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 01xx = 1/32
MUX 0	[3:0]	Selecciona el MUX para el Timer0. 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 01xx = 1/32

Table 5.4: Registro *TCON*.

Function	**Bits**	**Descripction**
Timer 5 auto reload on/off	[26]	Este bit determina el auto-reload para el Temporizador 5. 0 = One-shot 1 = Interval mode (auto reload)
Timer 5 manual update	[25]	Este bit determina el *manual update* del Temporizador 5. 0 = No operation 1 = Update TCNTB5
Timer 5 start/stop	[24]	Este bit determina el *start/stop* del Temporizador 5. 0 = Stop 1 = Start for Timer 5
Timer 4 auto reload on/off	[23]	Este bit determina el auto-reload para el Temporizador 4. 0 = One-shot 1 = Interval mode (auto reload)
Timer 4 output inverter on/off	[22]	Este bit determina el inversor de salida para el Temporizador 4. 0 = Inverter off 1 = Inverter on for TOUT4
Timer 4 manual update	[21]	Este bit determina el *manual update* del Temporizador 4. 0 = No operation 1 = Update TCNTB4, TCMPB4
Timer 4 start/stop	[20]	Este bit determina el *start/stop* del Temporizador 4. 0 = Stop 1 = Start for Timer 4
Timer 3 auto reload on/off	[19]	Este bit determina el auto-reload para el Temporizador 3. 0 = One-shot 1 = Interval mode (auto reload)
Timer 3 output inverter on/off	[18]	Este bit determina el inversor de salida para el Temporizador 3. 0 = Inverter off 1 = Inverter on for TOUT3
Timer 3 manual update	[17]	Este bit determina el *manual update* del Temporizador 3. 0 = No operation 1 = Update TCNTB3, TCMPB3
Timer 3 start/stop	[16]	Este bit determina el *start/stop* del Temporizador 3. 0 = Stop 1 = Start for Timer 3
Timer 2 auto reload on/off	[15]	Este bit determina el auto-reload para el Temporizador 2. 0 = One-shot 1 = Interval mode (auto reload)
Timer 2 output inverter on/off	[14]	Este bit determina el inversor de salida para el Temporizador 2. 0 = Inverter off 1 = Inverter on for TOUT2
Timer 2 manual update	[13]	Este bit determina el *manual update* del Temporizador 2. 0 = No operation 1 = Update TCNTB2, TCMPB2
Timer 2 start/stop	[12]	Este bit determina el *start/stop* del Temporizador 2. 0 = Stop 1 = Start for Timer 2
Timer 1 auto reload on/off	[11]	Este bit determina el auto-reload para el Temporizador 1. 0 = One-shot 1 = Interval mode (auto reload)
Timer 1 output inverter on/off	[10]	Este bit determina el inversor de salida para el Temporizador1. 0 = Inverter off 1 = Inverter on for TOUT1
Timer 1 manual update	[9]	Este bit determina el *manual update* del Temporizador 1. 0 = No operation 1 = Update TCNTB1, TCMPB1
Timer 1 start/stop	[8]	Este bit determina el *start/stop* del Temporizador 1. 0 = Stop 1 = Start for Timer 1
Dead zone enable	[4]	Este bit determina la operación de zona muerta. 0 = Disable 1 = Enable
Timer 0 auto reload on/off	[3]	Este bit determina el auto-reload para el Temporizador 0. 0 = One-shot 1 = Interval mode(auto reload)
Timer 0 output inverter on/off	[2]	Este bit determina el inversor de salida para el Temporizador 0. 0 = Inverter off 1 = Inverter on for TOUT0
Timer 0 manual update on/off	[1]	Este bit determina el *manual update* del Temporizador 0. 0 = No operation 1 = Update TCNTB0, TCMPB0
Timer 0 start/stop	[0]	Este bit determina el *start/stop* del Temporizador 0. 0 = Stop 1 = Start for Timer 0

Chapter 6

Matrix keyboard

6.1 Decription

The laboratory case contains a matrix keyboard. The keys in this type of keyboard are arranged in a two-dimensional array as shown in Figure 6.1: the keyboard is an array of horizontal and vertical lines with 16 switches (SB1-SB16) which connect a horizontal line with a vertical line when be pressed. Therefore, it is an extremely simple device, which requires additional logic to be used.

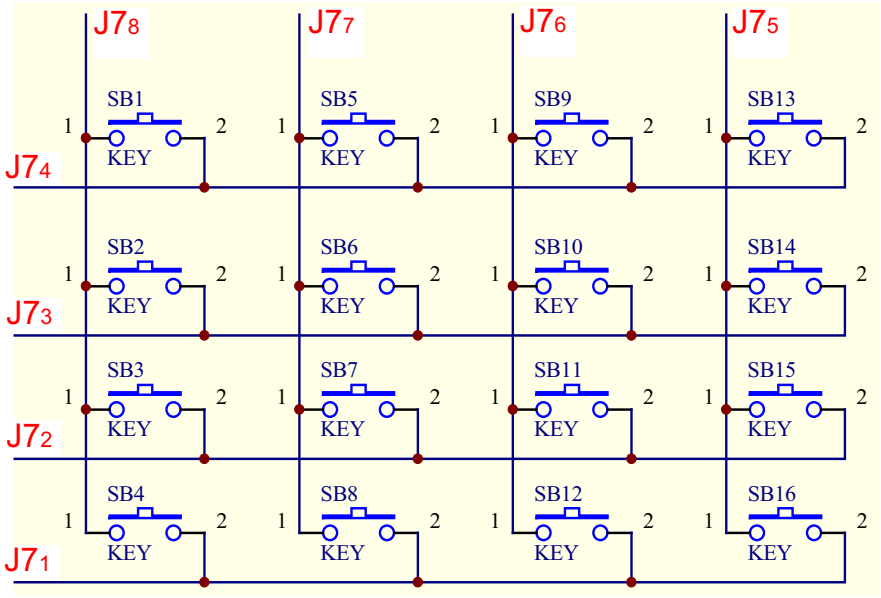

Figure 6.1: 4x4 keyboard circuit.

6.2 Keyboard Connection

This keyboard, located on a supplementary board, is connected to other components thorugh connector label as J7, it is located in the bottom right of the board. This connector has eight pins, and it is connected to the components shown in Figure 6.2. Unlike I/O controllers, this logic has no control/state/configuration or management registers, though simple, is unintuitive.

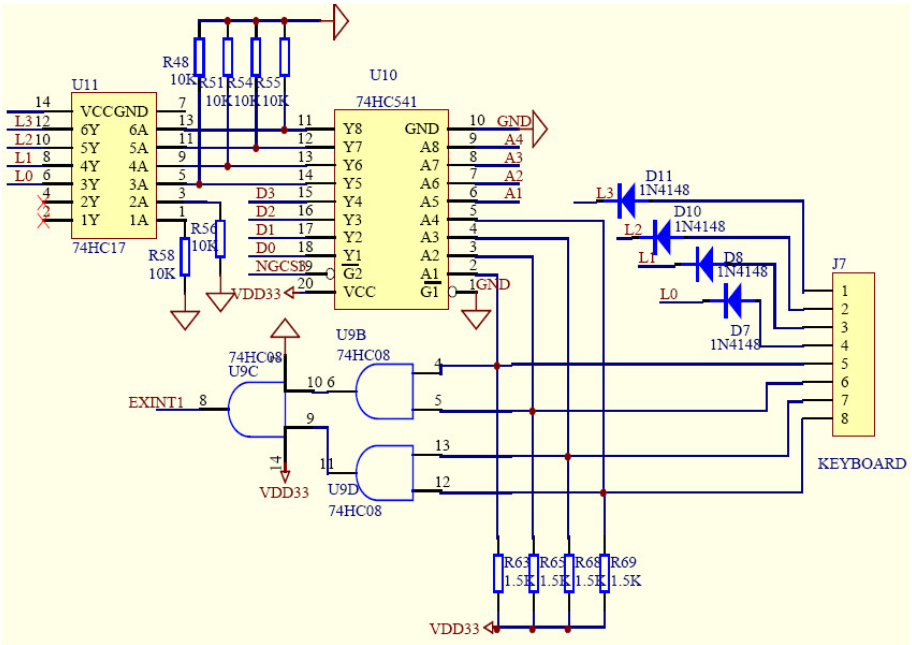

Figure 6.2: Keyboard Connection

The 5-8 pins of the connector 37 are associated with the columns of the keyboard (see Figure 6.1) and 1-4 pins are associated with the rows. Furthermore, pins 5-8 are connected to "*pull-up*" resistors, therefore in the absence of driver, the tension in them is high voltage level (Vdd33). These pins through the integrated circuit U9[1] generate the signal EINT1 of the interrupt controller. Also, these pins are connected to U10[2]. The outputs of this IC are connected to U11[3].

The 1-4 pins are connected to "*pull-down*" resistors, therefore in the absence of driver, the tension in them is low voltage level. There is not a direct connection, it is done through various components. The 1-4 pins are connected to the anode of four diodes (DN7/8/10/11)[4] and the cathodes of these devices are connected to the signals L0 -L3. Notice that these signals come from 3Y-6Y outputs of U11, and corresponding inputs (3A-6A) which are connected to "*pull-down*" resistors. Thus, when U10 is disabled (ie, when

[1] 74HC08: is an IC that contains AND gates implemented with CMOS technology, compatible with Schottky TTL technology.

[2] 74HC541A: contains an array of 8 tri-state inverting buffers designed for its use on buses. Internally, the output enable signal is an two complemented inputs AND gate ($\overline{G1}$ and $\overline{G2}$).

[3] 74HC17: an array of 6 non-inverting buffers designed to be used with diodes.

[4] 1N4148. Switching diodes for high speed applications

$NGCS3 = \overline{G2} = 1$) its Y1-Y8 outputs, be at high impedance and therefore 3A-6A inputs of U11 are at a low voltage due to the "*pull-down*" resistors.

Finally, Y1-Y4 pins of U10 are connected to D0-D3 lines of the data bus, while the A5-A8 pins of U10 are connected to the A1-A4 lines of the address bus.

6.3 Keyboard Operation

As mentioned in the previous section, while $NGCS3 = 1$, Y1-Y8 pins of U10 will keep at high impedance and 3A-6A pins of U11 will keep a low voltage due to the "*pull-down*" resistors. Consequently, L0-L3 lines be at low voltage, the DN7/8/10/11 diodes will be in conduction and 1-4 pins on J7 will be at low voltage. When a key is pressed, the switch connects the column associated with the corresponding row, so the column will be at low voltage. Then, one of the 5-8 pins will be at low voltage which, spreading through U9, will cause the EINT1 signal be at low level. If the G port and interrupt controller are configured properly (see section 3.2) this falling edge will trigger an IRQ interrupt.

Once the interruption has been detected, the interrupt service routine should identify the key pressed. In general, there are two ways:

- *Scanning*: This technique involves sending/placing a low level voltage on ONE horizontal line and a high voltage on the other of horizontal lines. When a vertical line be at low voltage ('0' logic level), then the key in the intersection of both lines, horizontal and vertical, has to be pressed key.

- *Inversion*: This technique involves sending a low voltage ('0' logic level) on the vertical lines and read the horizontal lines. If a horizontal line be at low level, it will indicate that a key has been pressed in that row. The procedure has to be repeated for every horizontal line, and read every one of the vertical lines. If a vertical line was at low voltage, then it indicates that a key has been pressed in that column. The key located at the intersection of the row and column has to be pressed key.

In practical sessions has be to used the "*scanning*" method, the only possible to be used on the S3CEV40 board.

To identify the column and row, and thus identify the key, we will proceed to the sequential sending a "0" for each one of the 1-4 pins of the J7 (while keeping with with ''1" value the remaining pins associated with rows) and read the 5-8 pins; when receive a "0" at any of them will mean that the row has electrical continuity with the receiving column, and therefore the key located at their intersection is the pressed key.

The sequential transmission of a "0" for each of the 1-4 pins of the J7 will be made sequentially writing the next patterns 4'b0111, 4'b1011, 4'b1101, 4'b1110 on A1-A4 lines in the address bus. For each of these patterns will proceed to read the 5-8 pins, whose values will be in the D0-D3 lines of the data bus. To make this possible, the values written in the address bus must be sent through U10. U10 enables its outputs when its two inputs are equal to zero$\overline{G1} = \overline{G2} = 0$, then must enable U10 making $nGCS3 = 0$. The signal $nGCS3 = 0$ is generated by the memory controller S3C44B0X when an address belonging to the range associated with bank-3, 0x0600_0000-0x07FF_FFFF is written to the address

bus. In other words, to write the pattern 4'b0111 on A4-A1 lines and spread on U10 is necessary to write to the address 0x0600_00EF in bus address; to write the pattern 4'b1011 should write the address 0x0600_00F7, and so on.

Once enabled the U10, the values written in the A1-A4 address bus lines will reach the 3Y-6Y output pins in the U11. When there is a "1" on one of the pins, the diode connected will cease to conduct and therefore the associated row is "floating" (it will have neither a voltage high VDD, or a voltage low, GND). If the pressed key is on that row, then there electrical continuity with the receiving column, but none of them (or row or column) is connected to any voltage source, so the value to be in the pin with J7 come imposed by the "*pull-up*" resistor. That is to say, on the data bus will be observed a level "1" on the corresponding line.

Instead, that diode connected to a pin of U11 on which the value is "0", will be conducting current, in such case if the key is pressed then pin at J7 associated with the corresponding column will take the value "0" and it will propagated to the data bus.

As summary, when sending sequentially only one "0" on each of the 1-4 pins at J7 (while keeping other 1-4 pins at "1") it must be observed the values in the data bus. The key is not identified, while the data bus reamins the value 0xF. The key is identified when there is a "0" value at any of the lines on the data bus, in fact, it means that are identified the row and column of the pressed key.

From the programmer's point of view, the previsous description can be considered as a read operation on the memory addresses 0x0600_00F7, 0x0600_00FB, 0x0600_00FD, 0x0600_00EF. For each one of these readings is necessary to identify if any of the return data 0-3 bits value is "0". When this happens have identified the key pressed.

Table 6.1 shows the values to be written in the address bus and read from the data bus when the corresponding key has been pressed. Keeping data 0xF on the data bus While the key no be identified.

Address	**Data**				
	0x7	0xB	0xD	0xE	0xF
0xFD	SB1	SB5	SB9	SB13	-
0xFB	SB2	SB6	SB10	SB14	-
0xF7	SB3	SB7	SB11	SB15	-
0xEF	SB4	SB8	SB12	SB16	-

Table 6.1: Correspondence between reading address and read data for each matrix keyboard key.

It is important to realize that during "*scanning*" process it will be generated a new interrupt request for EINT1. So, when writing a "0" on the row associated with the key pressed, this value will propagate through the column generating a "0" to any of the 5-8 pins on J7. This value will propagate through U9 generating a new transition to low on EINT1 and therefore a new interrupt request. This new application will not interrupt the execution of the RTI which is already running at that time because the ARM automatically disables the IRQ interrupts when servicing a IRQ interrupt (writting CSPR[7] = "1") but will be attended immediately upon completion of the current RTI (since leaving it is recovered the value that CSPR had before the RTI runs and therefore CSPR[7]="0"). Thus, a single

keystroke could launch several sequential executions of the same RTI. To avoid this problem the corresponding bit to EINT1 in the pending interrupts register (INTPND) should be deleted just before leaving the RTI.

Next, it is described an example of the key recognition method. When a key had been pressed which connects pins 1 and 5 of connector J7, and has been defined the routine treatment interruption. The RTI should write values to the address bus according to Table 6.1 and identifiying the key when reading the value indicated in that table from the four least significant bits of the data bus:

- Reading the contents of the memory address 0x0600_00FD (address within the range of addresses assigned to the keyboard; line A1 in the address bus set to "0", lines A2-A4 set to "1"). With this value on the address bus, the pin 1 of J7 is "floating" (not connected to any power line since it is cut diode D11). Since the pin 5 is connected to the pin 1, then the pin 5 is only connected to Vdd33 through a "*pull-up*" resistor. LAter, 5-8 pins are all on "1", the value of the four least significant bits of read data (from U10) is 0xF, and the signal $EINT1 = 1$.

- Reading the contents in the memory address 0x0600_00FB . Again, the value of the four least significant bits of read data (from U10) is 0xF, and the signal $EINT1 = 1$.

- Reading the contents of the memory address 0x0600_00F7. The value of the four least significant bits of read data (from U10) is 0xF, and the signal $EINT1 = 1$.

- Reading the contents of the memory address 0x0600_00EF. With this value the pin 1 from 37 is "0" (the diode D11 is conductive since its anode is on low voltage). Now, the pin 5 will have a "0", the signal $EINT1 = 0$ and data bus output is 0xE. This value matches with that given in Table 6.1. Then, the pressed key is the SB16.

  By identifying the key, and as a side effect, the EINT1 signal takes the value "0" creating a new interrupt request that should not be served because it is not due to any keystroke. Such request is not attended immediately, but is recorded in the register INTPND. Remind, it is necessary to clear the corresponding bit to EINT1 in the INTPND register just before the end of the RTI, to prevent re-run the RTI.

6.4 The bouncing Problem

When a key is pressed, does not produce a instant and stable transition in the voltage levels across the switch (it could be referred as "clean" transition), as expected in an ideal switching. This is due to the bouncing problem. Figure 6.3: a possible time diagram when pressing a key: flanks due to pressure and relasing a key are followed by a series of a bouncing in the voltage level.

To properly detect a pressed key is necessary to wait until the pressure bouncing finishes. One possible way consists of after detecting a key had been pressed (ie once serving the associated RTI), wait for more than the trp (pressure bouncing time) expected time, prior to identify the key pressed. For this type of keyboard we can assume $trp = 20ms$.

Also, it is also possible to eliminate the pressure bouncing . Since we not know the time when the user will stop pressing a key, the safest way to avoid the pressure bouncing

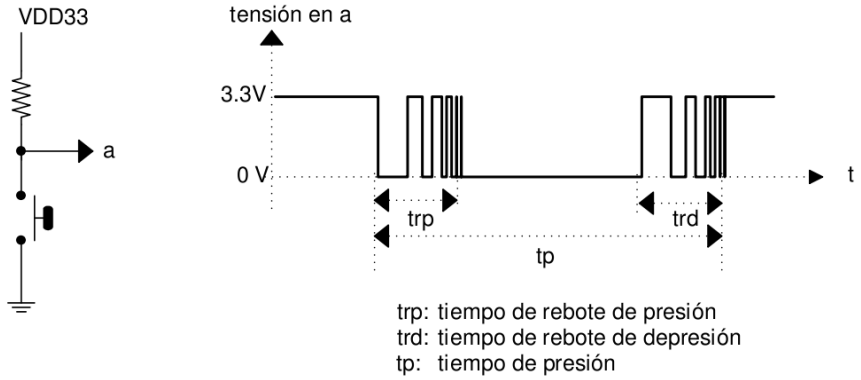

Figure 6.3: Bouncing phenomenon when pressing or releasing a key.

consists of detect the transition from the release, and only then wait for a larger time than the TRD (bounce time of depression), it will allow avoiding the realease bouncing before finish the identification process. To detect the release transition will wait for the line EINT1 takes the value "1" and then wait for $trd = 100ms$.

6.5 Keyboard, Interrupts and I/O pins

As we can see in Figure 6.2, the keyboard activates the EINT1 line when a key is pressed at repose state. As we saw previously, this line is externally accessible through the G port, which is how it connects to the external circuit U9B. To correctly generate the interrupt, the pin 1 of the port G must be configured to activate the EINT1 line when a falling edge is generated on such line.

Chapter 7

Serial Communication

Communication between two devices can be serial or parallel. Parallel communication uses various parallel wires to transmit a value of several bits simultaneously. Serial communication is characterized by information travels in only one wire, commonly referred as data line.

In serial communication, bits are injected into the data line and travel sequentially, one after another, from the source to the destination. Besides the data line may be other control and power lines. The serial communication can also be divided into two main groups:

- *Synchronous*. In synchronous serial communication both ends use a common clock signal. This signal is normally transmitted from the source to the receiver using/through an additional cable.
- *Asynchronous*. In asynchronous serial communication each system uses its own clock, although the two systems must keep the transfer rate.

Synchronous serial communication in the two systems are synchronized at the begginging of the transmission. Bits are detected on the flanks of the clock signal. Typically, it allows higher transfer rates but has the disadvantage that the clock signal should be transmitted, typically limiting the distance between the two computers.

The asynchronous serial communication has a dificulty on the detection/receiving correctly bits. Although, the two systems keep the same transmission frequency, it is virtually impossible keep same transmitting clock frequencies . First, both devices can work at different clock frequencies (eg PC and a serial printer) and can only generate from their onw clock, an approximate transmission speed signal (typically using fecuency dividers as those used in timers). Furthermore, these clocks can vary in frequency over time, for example by temperature variations. Small differences in the frequency eventually produce large time gaps in synchronized devices. The only one solution consists of limiting the number of bits that can transmited before force a new synchronization. That is the reason because many asynchronous serial protocols are limited to send small packages. These packets are called frames. At the beginning of each frame it is forced a synchronization between the two devices. To send multiple bytes is required to send multiple frames.

The usual implementation of a serial communication system builds a clock signal for transmission that be a multiple of the signal system, for example by dividing the system

frequency between 16 . It allows sampling the data line to detect the start of a frame. The gap since the frame is detected until the reception system starts will be at most one or two clock cycles of the system, which is a small gap with the transmission frequency.

7.1 UART unit in the S3C44BOX

The S3C44BOX system has two serial communication devices: a bus controller I^2C (synchronous) and a UART unit (*Universal Asynchronous Receiver and Transmitter*).

The UART unit provides two independent asynchronous serial ports: UART0 and UART1, also referred as channels, which allow a bidirectional serial communication up to 115.2 Kbps. The management of these channels can be carried out by *polling*, interruptions or DMA.

Each channel consists of the following elements (see Figure 7.1):

- a frequency generator (bauds).
- a transmitter module, with a shift and a buffer registers.
- a receiver module, with a shift and a buffer registers.
- a control unit.

The frequency generator, which defines the duration of each bit (and therefore the transfer rate) is used as the main input system clock (MCLK), whose frequency will be divided.

Transmission is through a shift register. When the processor wants to send a data on the line, the processor writes it in the transmission buffer. When the line is ready the register value is copied to the shift register. The content of this register will be shifted transfering sequentially its bits through the data line, on the speed indicated by the frequency generator. The reception is performed similarly by another shift register. The speed of transmission/reception is programmable by modifying the divider in the frequency generator.

In addition, both the transmitter and the receiver incorpore a 16 bytes FIFO queue. When active, these queues allow decouple the CPU and data communication through the serial port. Instead of writing/reading bytes one by one, the CPU writes/reads blocks up to 16 bytes. The contents of the FIFO queue is sent through the port (or receive and copy in the FIFO) automatically. The UART allows to activate an interrupt when the FIFO receiving queue exceeds a certain threshold (or emission buffer below a threshold), so the processor has some margin to address the situation.

Communication through the serial line is done byte by byte due to its asynchronous nature. Shift registers are responsible for transmitting or receiving bytes, bit by bit, the corresponding transmission TXDn line or RXDn reception line (where n represents the port number).

Finally, the S3C44BOX UART can transmit/receive infrared signals following the IrDA 1.0 standard.

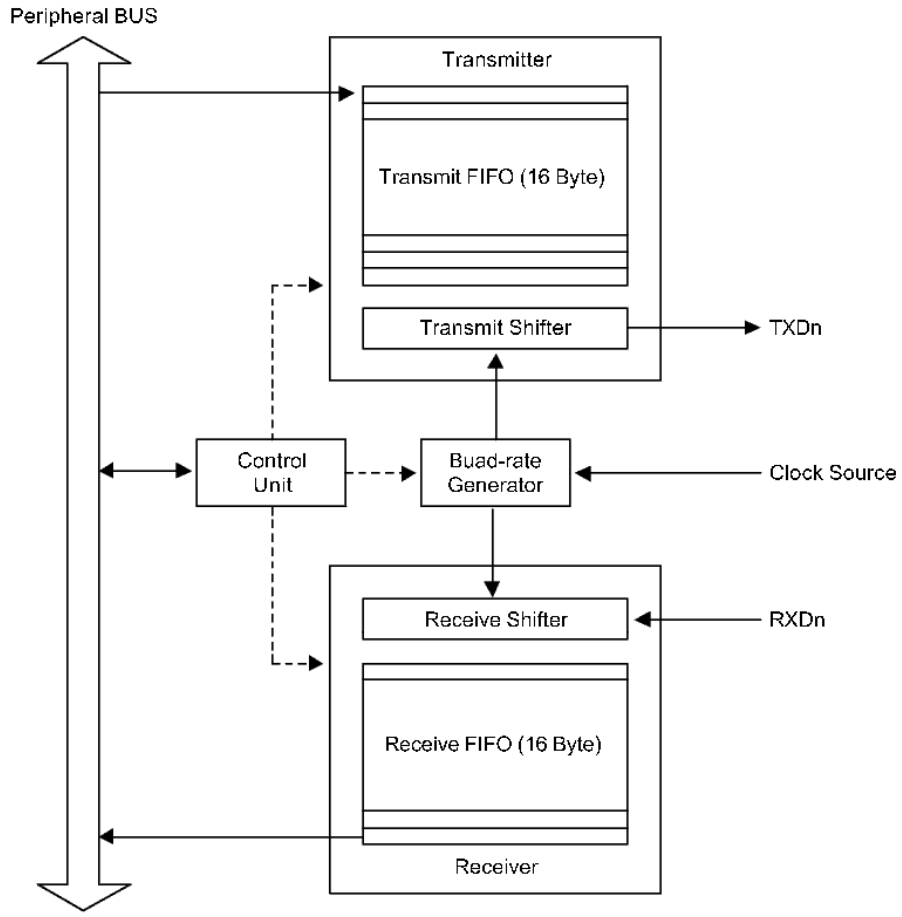

Figure 7.1: Estructura de un canal de la UART.

Next, the operation of the UART is briefly described. Detailed information can be found in Chapter 10 of the S3C44B0X Reference Manual [um-].

7.1.1 Frame Format

El formato de trama (*frame*) enviado por la UART tiene la siguiente estructura:

- Un primer bit de *start*. Sirve para que el receptor detecte el comienzo de la trama y se sincronice. Tiene el valor contrario al valor de reposo de la línea, en nuestro caso el bit de *start* es cero.
- 5-8 bits de datos.
- 1 bit de paridad (opcional). Se pueden escoger paridad par o impar.
- 1-2 bits de stop, que tienen el valor de reposo de la línea.

La figura 7.2 muestra un ejemplo de trama con el siguiente formato: 1 bit de *Start*, carácter de 8 bits, 1 bit de *Stop* y sin bit de paridad.

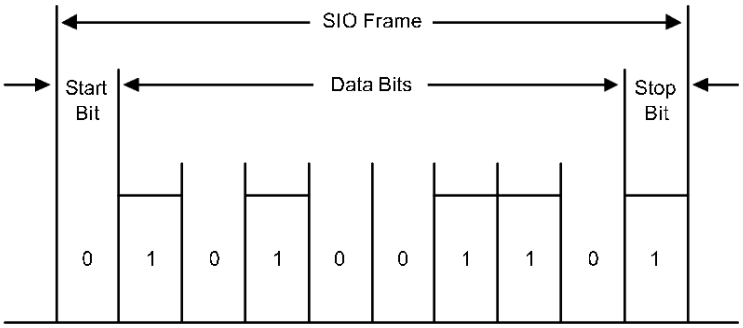

Figure 7.2: Diagrama temporal de una trama de datos serie (*SIO Frame*) con un tamaño de carácter de 8 bits, 1 bit de stop, sin paridad.

En la UART del S3C44BOX el formato de trama se configura mediante el registro de control de línea `UCONn` (`0x01D00000` y `0x01D04000`).

El transmisor también puede producir un *frame* de pausa o *break* que consiste en una señal de 0 lógico de un frame de duración. El propósito de este tipo de marco es informar al receptor de una pausa en la comunicación.

Al igual que para la transmisión, el formato de la trama en recepción también es configurable. Como es lógico, para una correcta comunicación es necesario que la configuración en ambos dispositivos sea la misma.

7.1.2 Errors

El receptor es responsable de detectar varios tipos de error de comunicación:

- Error de superposición. Indica que un dato ha sido sobrescrito por otro más reciente antes de ser leído.

- Error de paridad. Indica que la paridad del dato recibido no concuerda con la esperada.
- Error de trama. Indica que el dato recibido no tiene un bit de *Stop* válido.
- Solicitud de *break*. Indica que se ha recibido una trama de pausa (el valor de la línea se ha mantenido inactivo por un tiempo mayor que el de una trama).
- Error de *timeout* (sólo para gestión por DMA usando colas FIFO). Indica que ha transcurrido un intervalo de tiempo de 3 tramas sin recibir ningún dato, y el nivel de ocupación no llega al umbral fijado para la transmisión. Esto impide que el sistema se quede esperando por nuevos datos que quizá no vayan a llegar.

7.1.3 Flow Control

La UART del S3C44BOX admite control de flujo. En este caso el emisor debe esperar a que el receptor esté preparado para recibir. Esto se consigue añadiendo una línea adicional de control al puerto. Así la señal nCTS [1] (*Clear To Send*) del emisor se conecta con la señal nRTS (*Request To Send*) del receptor. Cuando éste está listo para recibir activa la señal nRTS. El emisor notará que se activa su señal nCTS y entonces podrá enviar el dato (ver figura 7.3).

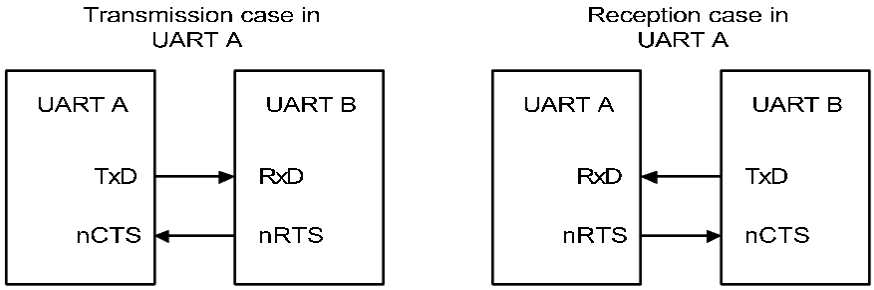

Figure 7.3: Conexión de dos UARTs empleando líneas de control de flujo.

En la UART del S3C44BOX la gestión de estas líneas puede llevarse a cabo de manera automática por HW (*Auto Flow Control*), o bien manualmente por SW, escribiendo en los registros de control y estado del módem (UMSTATn y UMCONn). Es preciso resaltar que el control de flujo automático sólo se puede emplear cuando se conectan dos UART entre sí. Para la conexión de un módem, es preciso usar el control SW, y, si se desea emplear un interfaz RS-232, es preciso usar además los puertos de E/S generales (ver figura 7.4).

7.1.4 Transfer speed on bauds

La velocidad de trabajo de la UART se establece dividiendo la señal de reloj principal (MCLK), 64MHz en nuestro caso, por un valor almacenado en el registro `UBRDIVn`. Dicho valor se obtiene mediante la fórmula:

$$UBRDIVn = (round\_off)(MCLK/(bps \times 16)) - 1$$

[1] Es preciso destacar que aquí la "n" indica que la señal se activa a baja.

El resultado de esta división debe estar comprendido entre 1 y $2^{16} - 1$.

Por ejemplo si la velocidad que se desea es de 115200 bps y la frecuencia de MCLK fuese de 40 MHz:

$$\begin{aligned} UBRDIVn &= (int)(40000000/(115200 \times 16) + 0.5) - 1 \\ &= (int)(21.7 + 0.5) - 1 \\ &= 22 - 1 = 21 \end{aligned}$$

7.1.5 Loop-Back Mode

La UART del S3C44BOX proporciona un modo de test que podríamos denominar de lazo cerrado o *loop-back*, cuyo propósito es ayudar a aislar errores en la comunicación y comprobar la corrección del SW. En este modo, el dato enviado es inmediatamente recibido por la propia UART.

La activación de este modo de test se realiza mediante un bit del registro de control de la UART (UCONn).

7.1.6 Settings of the UART

A continuación enunciaremos los principales registros de la UART, describiendo la funcionalidad de cada uno de ellos (para ampliar la descripción consultar el Manual de Referencia del S3C44BOX [um-]):

UART Line Control Register (ULCONn: 0x01D00000, 0x01D04000). Este registro se utiliza para la configuración de la línea de los puertos serie:

- Activar/desactivar modo IrDA.
- Configurar paridad (none/even/odd).
- Determinar bits de stop (1/2).
- Longitud del carácter (5/6/7/8).

La descripción del registro aparece en la tabla 7.1

UART Control Register (UCONn: 0x01D00004, 0x01D04004). Este registro se utiliza para configurar el funcionamiento de la UART:

- Tipo de interrupción de envío/recepción (pulso/flanco).
- Activar interrupción por timeout.
- Activar interrupción por error (*break*, *frame*, *parity*, *overrun*).
- Activar modo *loop-back*.
- Mandar señal de *break*.

Table 7.1: Registros de la UART ULCONn para configuración de la línea

ULCONn	Bit	Description
Reserved	[7]	
Infra-Red Mode	[6]	The Infra-Red mode determines whether or not to use the Infra-Red mode. 0 = Normal mode operation 1 = Infra-Red Tx/Rx mode
Parity Mode	[5:3]	The parity mode specifies how parity generation and checking are to be performed during UART transmit and receive operation. 0xx = No parity 100 = Odd parity 101 = Even parity 110 = Parity forced/checked as 1 111 = Parity forced/checked as 0
Number of stop bit	[2]	The number of stop bits specifies how many stop bits are to be used to signal end-of-frame. 0 = One stop bit per frame 1 = Two stop bit per frame
Word length	[1:0]	The word length indicates the number of data bits to be transmitted or received per frame. 00 = 5-bits 01 = 6-bits 10 = 7-bits 11 = 8-bits

- Determinar modo de transmisión para envío/recepción (desactivado, interrupción/encuesta o DMA).

La descripción del registro aparece en la tabla 7.2

UART FIFO Control Register (UFCONn: 0x01D00008, 0x01D04008). Estos registros se utilizan para la configuración de las colas FIFO de la UART:

- Habilitar FIFO.
- Determinar el nivel de ocupación con el que se desencadenan las interrupciones de envío/recepción FIFO.
- Borrar FIFO.

La descripción del registro aparece en la tabla 7.3

UART MODEM Control Register (UMCONn: 0x01D0000C, 0x01D0400C). Registro utilizado para configurar el control de flujo de la UART:

- Habilitar control de flujo automático (AFC).
- Activar la señal RTS (Ready To Send) cuando el control de flujo es SW.

La descripción del registro aparece en la tabla 7.4

Table 7.2: Registros UCONn para configuración del funcionamiento de la UART

UCONn	Bit	Description
Tx interrupt type	[9]	Interrupt request type 0 = Pulse (Interrupt is requested the instant Tx buffer becomes empty) 1 = Level (Interrupt is requested while Tx buffer is empty)
Rx interrupt type	[8]	Interrupt request type 0 = Pulse (Interrupt is requested the instant Rx buffer receives the data) 1 = Level (Interrupt is requested while Rx buffer is receiving data)
Rx time out enable	[7]	Enable/Disable Rx time out interrupt when UART FIFO is enabled. The interrupt is a receive interrupt. 0 = Disable 1 = Enable
Rx error status interrupt enable	[6]	This bit enables the UART to generate an interrupt if an exception, such as a break, frame error, parity error, or overrun error occurs during a receive operation. 0 = Do not generate receive error status interrupt 1 = Generate receive error status interrupt
Loop-back Mode	[5]	Setting loop-back bit to 1 causes the UART to enter the loop-back mode. This mode is provided for test purposes only. 0 = Normal operation 1 = Loop-back mode
Send Break Signal	[4]	Setting this bit causes the UART to send a break during 1 frame time. This bit is auto-cleared after sending the break signal. 0 = Normal transmit 1 = Send break signal
Transmit Mode	[3:2]	These two bits determine which function is currently able to write Tx data to the UART transmit holding register. 00 = Disable 01 = Interrupt request or polling mode 10 = BDMA0 request (Only for UART0) 11 = BDMA1 request (Only for UART1)
Receive Mode	[1:0]	These two bits determine which function is currently able to read data from UART receive buffer register. 00 = Disable, 01 = Interrupt request or polling mode 10 = BDMA0 request (Only for UART0) 11 = BDMA1 request (Only for UART1)

Table 7.3: Registros UFCONn para configuración de las colas FIFO de la UART

UFCONn	Bit	Description	Initial State
Tx FIFO Trigger Level	[7:6]	These two bits determine the trigger level of transmit FIFO. 00 = Empty 01 = 4-byte 10 = 8-byte 11 = 12-byte	00
Rx FIFO Trigger Level	[5:4]	These two bits determine the trigger level of receive FIFO. 00 = 4-byte 01 = 8-byte 10 = 12-byte 11 = 16-byte	00
Reserved	[3]		0
Tx FIFO Reset	[2]	This bit is auto-cleared after resetting FIFO 0 = Normal 1= Tx FIFO reset	0
Rx FIFO Reset	[1]	This bit is auto-cleared after resetting FIFO 0 = Normal 1= Rx FIFO reset	0
FIFO Enable	[0]	0 = FIFO disable 1 = FIFO mode	0

Table 7.4: Registros UFCONn para configurar el control de flujo de la UART

UMCONn	Bit	Description
Reserved	[7:5]	These bits must be 0's
AFC(Auto Flow Control)	[4]	0 = Disable 1 = Enable
Reserved	[3:1]	These bits must be 0's
Request to Send	[0]	If AFC bit is enabled, this value will be ignored. In this case the S3C44B0X will control nRTS automatically. If AFC bit is disabled, nRTS must be controlled by S/W. 0 = 'H' level(Inactivate nRTS) 1 = 'L' level(Activate nRTS)

Table 7.5: Registros UTRSTAT con información del estado de la transmisión y la recepción

UTRSTATn	Bit	Description
Transmit shifter empty	[2]	This bit is automatically set to 1 when the transmit shift register has no valid data to transmit and the transmit shift register is empty. 0 = Not empty 1 = Transmit holding & shifter register empty
Transmit buffer empty	[1]	This bit is automatically set to 1 when the transmit buffer register does not contain valid data. 0 =The buffer register is not empty 1 = Empty If the UART uses the FIFO, users should check Tx FIFO Count bits and Tx FIFO Full bit in the UFSTAT register instead of this bit.
Receive buffer data ready	[0]	This bit is automatically set to 1 whenever the receive buffer register contains valid data, received over the RXDn port. 0 = Completely empty 1 = The buffer register has a received data If the UART uses the FIFO, users should check Rx FIFO Count bits in the UFSTAT register instead of this bit.

UART Tx/Rx Status Register (UTRSTATn: 0x01D00010, 0x01D04010). Registros de estado de transmisión y recepción, utilizados para:

- Determinar si el shifter de transmisión está vacío.
- Determinar si el buffer de transmisión está vacío.
- Determinar si el buffer de recepción tiene datos listos.

La descripción del registro aparece en la tabla 7.5.

UART Rx Error Status Register (UERSTATn: 0x01D00014, 0x01D04014). Determina el tipo de error que ha desencadenado la interrupción por error en la recepción. Su descripción aparece en la tabla 7.6.

Table 7.6: Registros UERSTAT con información del estado de los errores en la recepción.

UERSTATn	Bit	Description
Break Detect	[3]	This bit is automatically set to 1 to indicate that a break signal has been received. 0 = No break receive 1 = Break receive
Frame Error	[2]	This bit is automatically set to 1 whenever a frame error occurs during receive operation. 0 = No frame error during receive 1 = Frame error
Parity Error	[1]	This bit is automatically set to 1 whenever a parity error occurs during receive operation. 0 = No parity error during receive 1 = Parity error
Overrun Error	[0]	This bit is automatically set to 1 whenever an overrun error occurs during receive operation. 0 = No overrun error during receive 1 = Overrun error

Table 7.7: Registros UFSTAT con información del estado de las colas FIFO de la UART.

UFSTATn	Bit	Description
Reserved	[15:10]	
Tx FIFO Full	[9]	This bit is automatically set to 1 whenever transmit FIFO is full during transmit operation 0 = 0-byte ≤ Tx FIFO data ≤ 15-byte 1 = Full
Rx FIFO Full	[8]	This bit is automatically set to 1 whenever receive FIFO is full during receive operation 0 = 0-byte ≤ Rx FIFO data ≤ 15-byte 1 = Full
Tx FIFO Count	[7:4]	Number of data in Tx FIFO
Rx FIFO Count	[3:0]	Number of data in Rx FIFO

UART FIFO Status Registers (UFSTATn: 0x01D00018, 0x01D04018). Registros de estado para las colas FIFO de la UART, utilizados para:

- Determinar si el FIFO de envío/recepción está lleno/vacío.
- Determinar el nº de elementos presentes en la cola FIFO.

La descripción del registro aparece en la tabla 7.7.

UART Modem Status Register (UMSTATn: 0x01D0001C, 0x01D0401C). Gestión de la señal CTS (*Clear To Send*) en el control de flujo SW. Su descripción aparece en la tabla 7.8.

Table 7.8: Registros UMSTAT para observar la línea CTS en el control de flujo software.

UMSTATn	Bit	Description
Delta CTS	[4]	This bit indicates that the nCTS input to S3C44B0X has changed state since the last time it was read by CPU. (Refer to Fig. 10-7) 0 = Has not changed 1 = Has changed
Reserved	[3:1]	Reserved
Clear to Send	[0]	0 = CTS signal is not activated(nCTS pin is high) 1 = CTS signal is activated(nCTS pin is low)

Table 7.9: Direcciones de los registros URXH.

Register	Address	R/W	Description	Reset Value
URXH0	0x01D00024(L) 0x01D00027(B)	R (by byte)	UART channel 0 receive buffer register	–
URXH1	0x01D04024(L) 0x01D04027(B)	R (by byte)	UART channel 1 receive buffer register	–

UART Transmit Holding Register (UTXHn) y UART Receive Holding Register (URXHn: UTXH0, UTXH1). UTXHn son los registros en los que debemos escribir el dato que se desea transmitir. En URXHn podremos leer el dato recibido. Debemos resaltar que cuando se produce un error de superposición (*overrun*) se debe leer URXHn o de lo contrario, al recibir el siguiente dato, se volverá a producir un nuevo error. Las direcciones de estos registros están descritas en la tabla 7.9 y 7.10:

UART Baud Rate Division Register (UBRDIV: 0x01D00028, 0x01D04028). Permite establecer la frecuencia de comunicación. La fórmula para calcular el ratio y la velocidad en baudios se ha explicado anteriormente en la sección 7.1.4.

7.1.7 Connecting the UART serial ports in the trainning board

La figura 7.4 muestra cómo se establece la conexión entre el S3C44BOX y los conectores DB9 de la placa S3CEV40. Como puede apreciarse en la figura, el conector UART0 dispone únicamente de dos líneas (RXD y TXD) y por lo tanto sólo puede emplearse para comunicaciones serie simples (sin control de flujo). El conector UART1 además de las líneas de transmisión y recepción, está conectado a 6 pines de E/S generales lo que permite que, con una gestión adecuada de los mismos, se pueda conectar a un modem RS-232.

Table 7.10: Direcciones de los registros UTXH.

Register	Address	R/W	Description
UTXH0	0x01D00020(L) 0x01D00023(B)	W (by byte)	UART channel 0 transmit holding register
UTXH1	0x01D04020(L) 0x01D04023(B)	W (by byte)	UART channel 1 transmit holding register

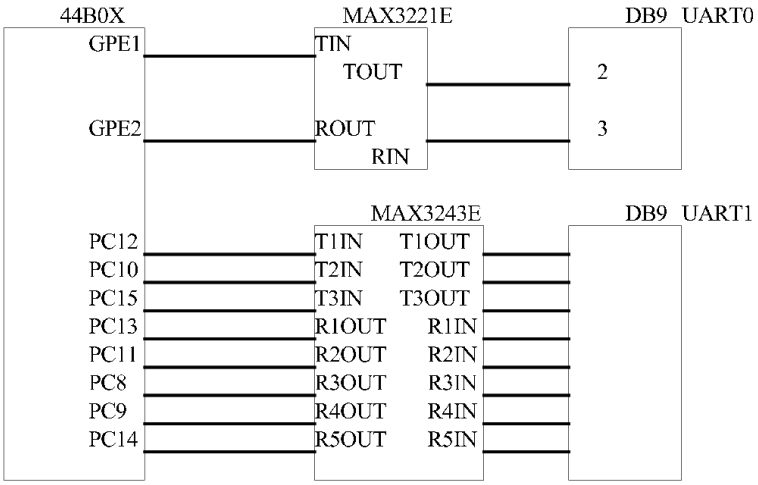

Figure 7.4: Conexión de la UART a los puertos serie de la placa (DB9).

Chapter 8

DMA controller

El S3C44B0X incorpora un controlador de *acceso directo a memoria* (DMA), que permite realizar transferencias memoria a memoria, memoria a periférico y periférico a memoria sin intervención del procesador. El controlador de DMA se encarga de solicitar el bus y luego controlarlo para que las transferencias se realicen de forma correcta.

El controlador de DMA del sistema tiene cuatro canales. Dos de ellos, ZDMA0 y ZDMA1, están conectados al bus de sistema. Los otros dos, BDMA0 y BSMA1, están localizados en el *bridge* que hace de interfaz con el bus de periféricos.

Los canales ZDMAn se utilizan para realizar transferencias de datos desde memoria a memoria, o entre la memoria y dispositivos de E/S mapeados en direcciones fijas de memoria. Los canales BDMAn están orientados a transferencias entre la memoria y los dispositivos de entrada/salida conectados al bus de periféricos, como SIO, IIS, timers o UART.

En la práctica vamos a realizar transferencias DMA para enviar datos del ARM al PC a través del puerto serie, es decir, a través de la UART. Por ello vamos a utilizar el controlador BDMA, cuyo diagrama aparece en la Figura 8.1. En esta sección describiremos en detalle el controlador BDMA, para ampliar la información sobre los controladores de DMA se aconseja consultar el manual del S3C44B0X [um-].

Aunque el BDMA puede ser utilizado para realizar transferencias memoria a memoria, es preferible utilizar para ello el ZDMA que tiene un buffer temporal para almacenar bloques de 16 bytes pudiendo realizar transferencias en modo *burst*.

El controlador BDMA sólo admite un protocolo de comunicación, *handshake* y un modo de transferencia, *Unit transfer*. El primero define cómo se activan/desactivan las señales de petición, *request*, y reconocimiento, *acknowledge*, en el bus. El segundo define la granularidad con la que se realizan las operaciones de lectura-escritura. Concretamente en el modo *Unit transfer* el controlador solicita el bus para realizar un par lectura-escritura y luego lo libera, como se indica en la figura 8.2. Cada lectura-escritura puede ser de tamaño byte, media palabra o palabra. El par lectura-escritura no puede ser interrumpido. Si el controlador de DMA tiene que hacer más transferencias vuelve a solicitar el bus.

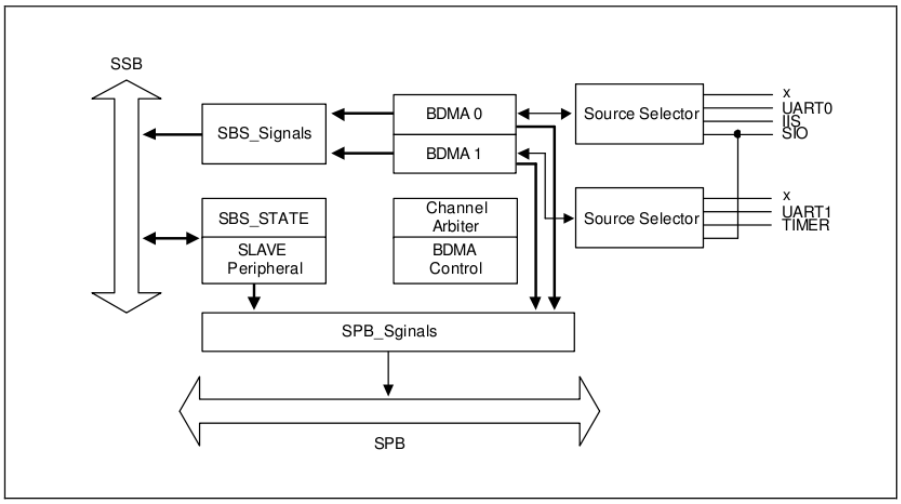

Figure 8.1: Diagrama de bloques del controlador BDMA.

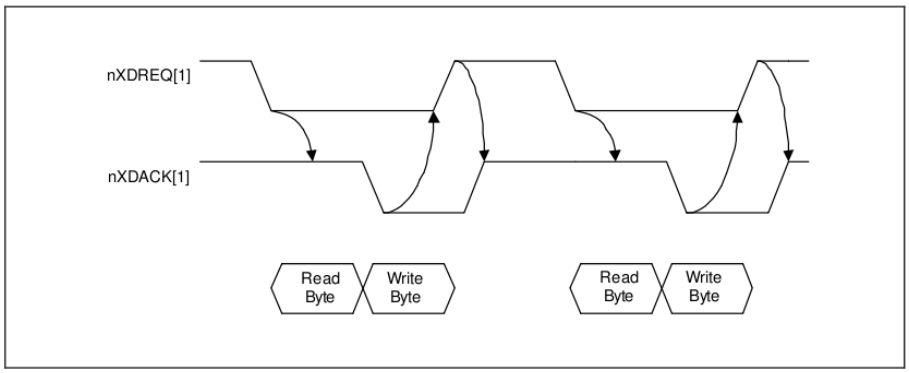

Figure 8.2: Cronograma del modo de *handshake-Unit transfer* utilizado por el controlador BDMA.

8.1 Programming DMA transfers

Programar una transferencia DMA es sencillo. Para ello debemos configurar el controlador DMA correspondiente con la dirección origen, la dirección destino y el número de bytes a transferir. Después, si la señalización es software debemos activar la petición DMA escribiendo en el registro correspondiente. Si la señalización es hardware la transferencia se realizará cuando el periférico active la petición. Además, de forma similar a como sucedía con los temporizadores, el controlador de DMA tiene un sistema de doble buffer. Al inicio, los registros de inicialización de direcciones y el contador de bytes a transferir se copian en los registros de trabajo. Mientras se realiza una transferencia DMA podemos programar la siguiente modificando los registros de inicialización. Debemos tener en cuenta que no basta con escribir los valores correctos en estos registros, el controlador no realizará la transferencia hasta que el periférico la solicite (asumiendo señalización hardware).

8.2 DMA Controller Configuration

Como el resto de los controladores que hemos visto, el controlador de DMA se configura a través de una serie de registros mapeados en memoria, que son:

- *BDCONn (0x01F80000, 0x01F80020)* Registro que nos permite comprobar el estado de la transferencia en curso, si ha terminado o no, habilitar o no las peticiones y realizar la señalización software. La descripción de estos registros se recoge en la figura 8.3.

BDCONn	Bit	Description	Initial State
INT	[7:6]	Reserved	00
STE	[5:4]	Status of DMA channel (Read only) 00 = Ready 01 = Not TC yet 10 = Terminal Count 11 = N/A Before the DMA counter decreases from a initial counter value, STE is still the ready state.	00
QDS	[3:2]	Disable/Enable External/Internal DMA request (UARTn, SIO, IIS, Timer) 00 = Enable Other = Disable	00
CMD	[1:0]	Software commands 00: No command. After writing 01, 10, 11, CMD bits are cleared automatically. 01: Reserved 10: Reserved 11: Cancels DMA operation.	00

Figure 8.3: Registros de control del controlador BDMA.

- *BDISRCn (0x01F80004, 0x01F80024) y BDCSRCn (0x01F80010,0x01F80030)* Sirven para especificar la dirección origen de la transferencia, si hay que incrementar o decrementar la dirección para completar la transferencia y el tamaño (byte, media palabra o palabra). El de inicialización (BDISRC) se copia en el actual (BDCSRC) al comienzo. La descripción de estos registros se da en la figura 8.4.

- *BDIDESn (0x01F80008, 0x01F80014) y BDCDESn (0x01F80028, 0x01F80034)* Registros para configurar la dirección destino de la transferencia, la dirección (memoria

BDISRCn/BDCSRCn	Bit	Description	Initial State
DST	[31:30]	Data size for transfer 00 = Byte 01 = Half word 10 = Word 11 = Not used	00
DAL	[29:28]	Direction of address for load 00 = N/A 01 = Increment 10 = Decrement 11 = Internal peripheral (fixed address)	00
ISADDR/CSADDR	[27:0]	Initial/current source address for BDMAn. If the destination is the internal peripherals, the SFR address has to be used. For example, if the source is the UART0 Rx buffer, the UART0 Rx buffer address will be used.	0x0000000

Figure 8.4: Registros de configuración del origen de la transferencia.

a periférico, periférico a memoria o periférico a periférico) y modo en que hay que cambiar la dirección para completar la transferencia (incrementar o decrementar). La descripción de estos registros se da en la figura 8.5.

BDIDESn/BDCDESn	Bit	Description	Initial State
TDM	[31:30]	Transfer direction mode 00 = Reserved 01 = M2IO (from external memory to internal peripheral) 10 = IO2M (from internal peripheral to external memory) 11 = IO2IO (from internal peripheral to internal peripheral) **NOTE:** The initial value is '00', but you must change TDM value as another though the BDMA channel is unused.	00
DAS	[29:28]	Direction of address for store 00 = N/A 01 = Increment 10 = Decrement 11 = Internal peripheral (fixed address)	00
IDADDR/CDADDR	[27:0]	Initial/current destination address for BDMAn If the destination is the internal peripherals, the SFR address has to be used. For example, if the destination is UART0 Tx buffer, the UART0 Tx buffer address will be used.	0x0000000

Figure 8.5: Registros de configuración del destino de la transferencia.

- *BDICNT (0x01F8000C, 0x01F80018) y BDCCNT (0x01F8002C, 0x01F80038)* Permiten configurar el número de bytes a transmitir, habilitar o deshabilitar el propio DMA, activar el modo auto-reload, seleccionar la fuente y el modo de interrupciones. La descripción de estos registros aparece en la figura 8.6.

BDICNT0/BDCCNT0	Bit	Description	Initial State
QSC	[31:30]	DMA request source selection 00 = N/A 01 = IIS 10 = UART0 11 = SIO	00
Reserved	[29:28]	00: handshake mode	00
Reserved	[27:26]	01: unit transfer mode	01
Reserved	[25:24]	00: on-the-fly mode is not supported in BDMAn	00
INTS	[23:22]	Interrupt mode set 00 = Polling mode 01 = N/A 10 = Int. whenever transferred 11 = Int. whenever terminated count	00
AR	[21]	Auto-reload and Auto-start after DMA count are 0. 0= Disable 1= Enable. Even after DMA count is 0, the DMA H/W enable bit (EN bit) is still 1. But, DMA will start to operate only if the start command or DMA request is activated	0
EN	[20]	DMA H/W enable/disable 0 = Disable DMA 1 = Enable DMA. If the QDS bit is 00b, DMA request can be serviced. Also if the S/W command is started, the DMA operation will occur. If the EN bit is 0, DMA will not operate even though S/W command is started. If the S/W command is canceled, the DMA operation will be canceled and EN bit will be cleared to 0. At the terminal count, the EN bit will be cleared to 0. **NOTE:** Do not set the EN bit and the other bits of BDICNT register at the same time. User have to set EN bit after setting the other bits of BDICNT register as following steps, 1. Set BDICNT register with disabled En bit. 2. Set EN bit enable.	0
ICNT/CCNT	[19:0]	Transfer count for BDMA0. The transfer count must be right value. For example, if DST is word, ICNT must be 4n. If 1 byte is transferred, the ICNT will be decreased by 1. If 1 half-word is transferred, the ICNT will be decreased by 2. If 1 word is transferred, the ICNT will be decreased by 4.	0x00000

Figure 8.6: Registros de configuración de la cuenta.

Bibliography

[um-] S3c44b0x risc microprocessor product overview. Accesible en http://www.samsung.com/global/business/semiconductor/productInfo.do?fmly_id=229&partnum=S3C44B0. Hay una copia en el campus virtual.