4/28/2011

CALVIN
HOLIC

DIGITAL ALARM CLOCK DESIGN

CSE379 | Design Project

Introduction

The following is a design for an ARM microprocessor-based digital alarm clock. The system has 4MB of RAM, 4MB of ROM, and 5KB of memory-mapped I/O. Peripherals include 3 PrimeCell General Purpose Input/Output (GPIO) units, a PrimeCell Universal Asynchronous Receiver/Transmitter (UART) and a PrimeCell Real Time Clock (RTC) with an attached keypad.

The non-PrimeCell components are as follows:

- 4 Seven-Segment Displays – For displaying the time
- 3 LEDs
    - 2 for the colon separating the two hour digits from the two minute digits
    - 1 to indicate PM time
- 4 Momentary Push Buttons
    - Set Time Button
    - Set Alarm Button
    - Turn Off Alarm Button
    - Snooze Button
- 1 Buzzer – For the actual alarm signal
- 1 Keypad – For entering time information into the UART

For these components, I've selected the following parts, with accompanying datasheet links:

- **Seven-Segment Display:** Lite-On Electronics, Inc. Part no. LSHD-7803
    - http://media.digikey.com/pdf/Data%20Sheets/Lite-On%20PDFs/LSHD-7803.pdf
- **LEDs:** Lite-On Electronics, Inc. Part no. LTST-C190
    - http://media.digikey.com/pdf/Data%20Sheets/Lite-On%20PDFs/LTST-C190TBKT.pdf
- **Push Buttons:** Panasonic Part no. ESE20C
    - http://industrial.panasonic.com/www-data/pdf/ATU0000/ATU0000CE7.pdf
- **Buzzer:** Mallory Sonalert Products Inc. Part no. ASI301Q
    - http://www.mallory-sonalert.com/specifications/ASI301Q.pdf
- **Keypad:** Grayhill Inc. Part no. 96AB2-102-F
    - http://media.digikey.com/pdf/Data%20Sheets/Grayhill%20PDFs/96%20Series.pdf
- **Keypad Encoder:** Fairchild Semiconductor Part no. MM74C922N
    - http://media.digikey.com/pdf/Data%20Sheets/Fairchild%20PDFs/MM74C922,923.pdf
- **Parallel to Serial Converter:** Micrel Part no. SY10E446JC
    - http://www.micrel.com/_PDF/HBW/sy10-100e446.pdf

Memory

Below is a list of system components requiring memory, and the amount of memory needed for each unit. The number of address lines needed to cover this size is found by taking the $\log_2$ of the required size.

- ROM
  - 4MB Required
  - 22 Address Lines
- RAM
  - 4MB Required
  - 22 Address Lines
- GPIO (3)
  - Memory addressing ranges from 0x00 to 0xFF for PrimeCell GPIO, as found in the ARM PrimeCell Technical Reference Manual. Therefore, $2^8 = 256B$ is necessary for each GPIO unit
  - 768B Required Total
  - 8 Address Lines per unit
- UART
  - The PrimeCell UART unit also ranges from 0x00 to 0xFF, resulting in 256B
  - 256B Required
  - 8 Address Lines
- RTC
  - For the PrimeCell RTC, addressing ranges from 0x000 to 0xFFC. Filling out the last 2 addresses to 0xFFF, $2^{12} = 4KB$ is required for the Real Time Clock
  - 4KB Required
  - 12 Address Lines

In order to address all of these components in one memory system, partially decoded memory addressing is used. With respect to RAM-ROM Memory versus Memory-Mapped I/O, address lines A[21:0] are used in addressing with lines A[22] and A[23] used as selector bits as described below:

A[23]	A[22]	Memory Selection
0	0	ROM
0	1	RAM
1	0	I/O
1	1	Unused

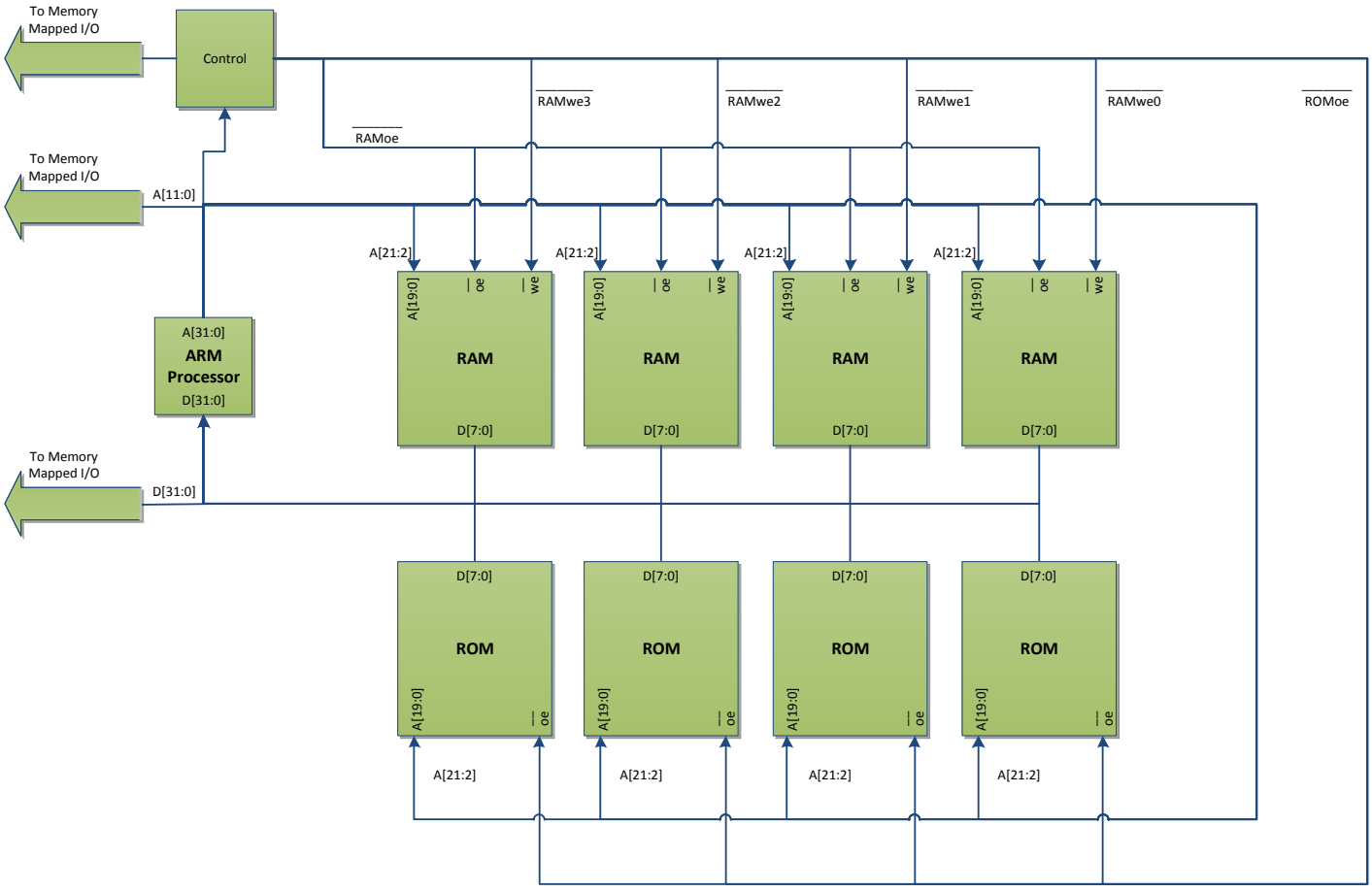
Within I/O, less than 22 address lines are required, but more selecting bits are required. Since the RTC uses the largest amount of memory with 12 address lines A[11:0], the 13th address line A[12] will be used as a selector bit:

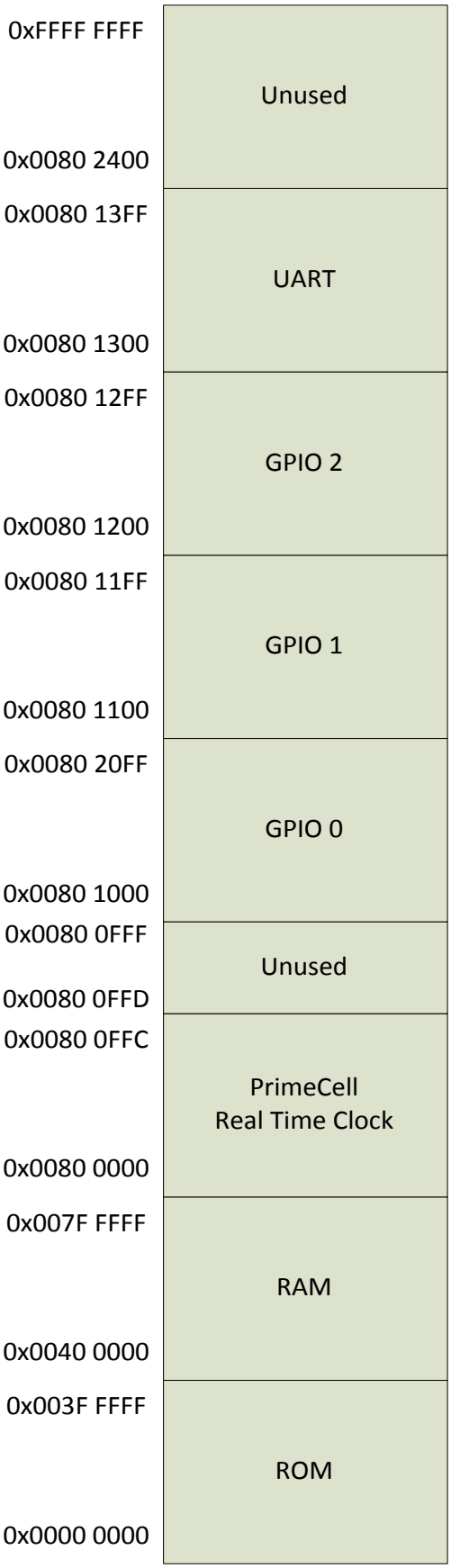
A[12]	Memory Selection
0	RTC
1	GPIO/UART

And lastly, for GPIO/UART selection, the following decoding scheme is used:

A[9]	A[8]	Memory Selection
0	0	GPIO 0
0	1	GPIO 1
1	0	GPIO 2
1	1	UART

The resulting memory map is shown on the following page.





This partial decoding scheme results in contiguous memory blocks for all required memory (save the 3 addresses unused at the back end of the RTC).

System Overview

Below is an overview of the Advanced Microcontroller Bus Architecture (AMBA) compliant system design. The ARM microprocessor, memory units, and bridge are connected via the Advanced High-Performance Bus for high performance data transfer, while the I/O units are connected to the bridge via the Advanced Peripheral Bus.

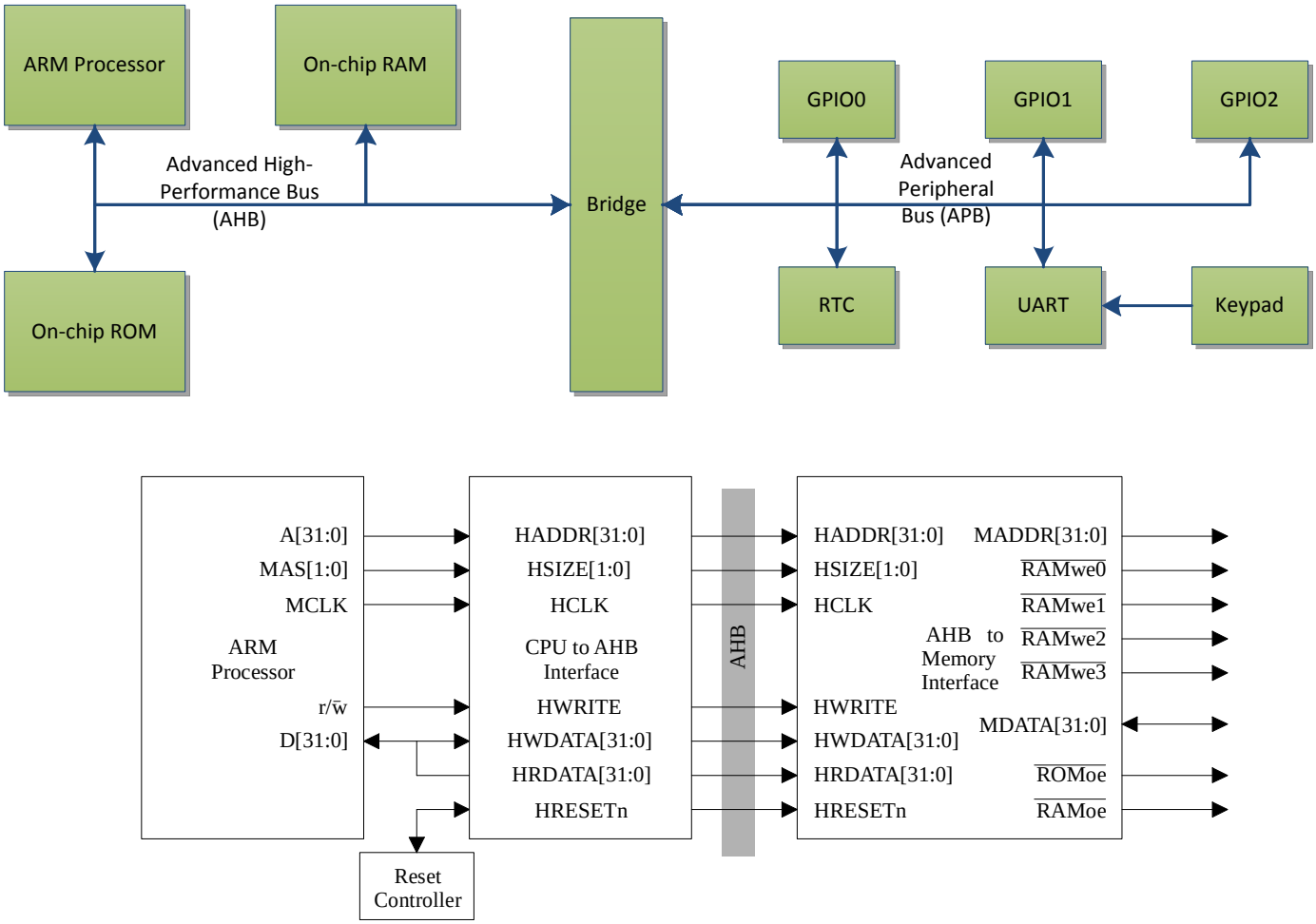
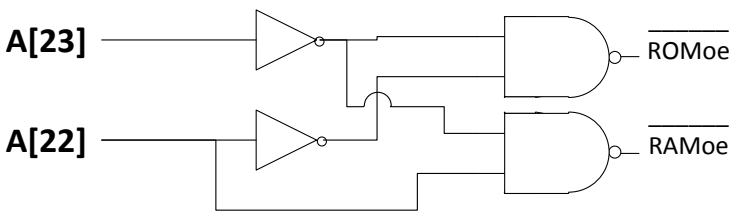


Figure from slide 5, R. Sridhar, *CSE 379 Lecture 10 (AMBA Interface/GPIO)*, University at Buffalo, Spring 2007

The figure above shows the interface between memory and the AHB. Within the right-most "Control" section, the following decoder circuit (appropriate for the previously presented addressing scheme) is present for RAM and ROM selection:



The truth table governing the control unit is as follows:

Operation	A23	A22	A12	A9	A8	MAS[1]	MAS[0]	A1	A0	R'/W	ROMoe'	RAMoe'	RAMwe0'	RAMwe1'	RAMwe2'	RAMwe3'
ROM Read	0	0	-	-	-	-	-	-	-	0	0	1	1	1	1	1
RAM Read	0	1	-	-	-	-	-	-	-	0	1	0	1	1	1	1
RAM Write (Word)	0	1	-	-	-	1	0	0	0	1	1	1	0	0	0	0
RA M Write	0	1	-	-	-	0	1	0	0	1	1	1	0	0	1	1
(Halfword)	0	1	-	-	-	0	1	1	0	1	1	1	1	1	0	0
RAM Write (Byte)	0	1	-	-	-	0	0	0	0	1	1	1	0	1	1	1
	0	1	-	-	-	0	0	0	1	1	1	1	1	0	1	1
	0	1	-	-	-	0	0	1	0	1	1	1	1	1	0	1
	0	1	-	-	-	0	0	1	1	1	1	1	1	1	1	0
RTC Read	1	0	0	-	-	-	-	-	-	0	1	1	1	1	1	1
GPIO0 Read	1	0	1	0	0	-	-	-	-	0	1	1	1	1	1	1
GPIO1 Read	1	0	1	0	1	-	-	-	-	0	1	1	1	1	1	1
GPIO2 Read	1	0	1	1	0	-	-	-	-	0	1	1	1	1	1	1
UART Read	1	0	1	1	1	-	-	-	-	0	1	1	1	1	1	1
RTC Write	1	0	0	-	-	-	-	-	-	1	1	1	1	1	1	1
GPIO0 Write	1	0	1	0	0	-	-	-	-	1	1	1	1	1	1	1
GPIO1 Write	1	0	1	0	1	-	-	-	-	1	1	1	1	1	1	1
GPIO2 Write	1	0	1	1	0	-	-	-	-	1	1	1	1	1	1	1
UART Write	1	0	1	1	1	-	-	-	-	1	1	1	1	1	1	1

The decoding required for the memory-mapped I/O is contained in the APB Bridge. Below are figures of the APB Bridge, the truth table for its selection control bits, and the corresponding memory address decoding circuit.

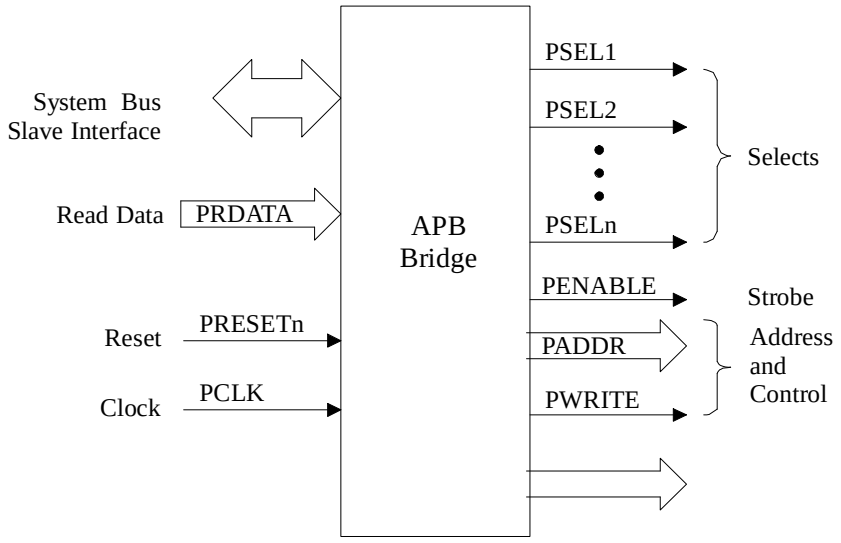
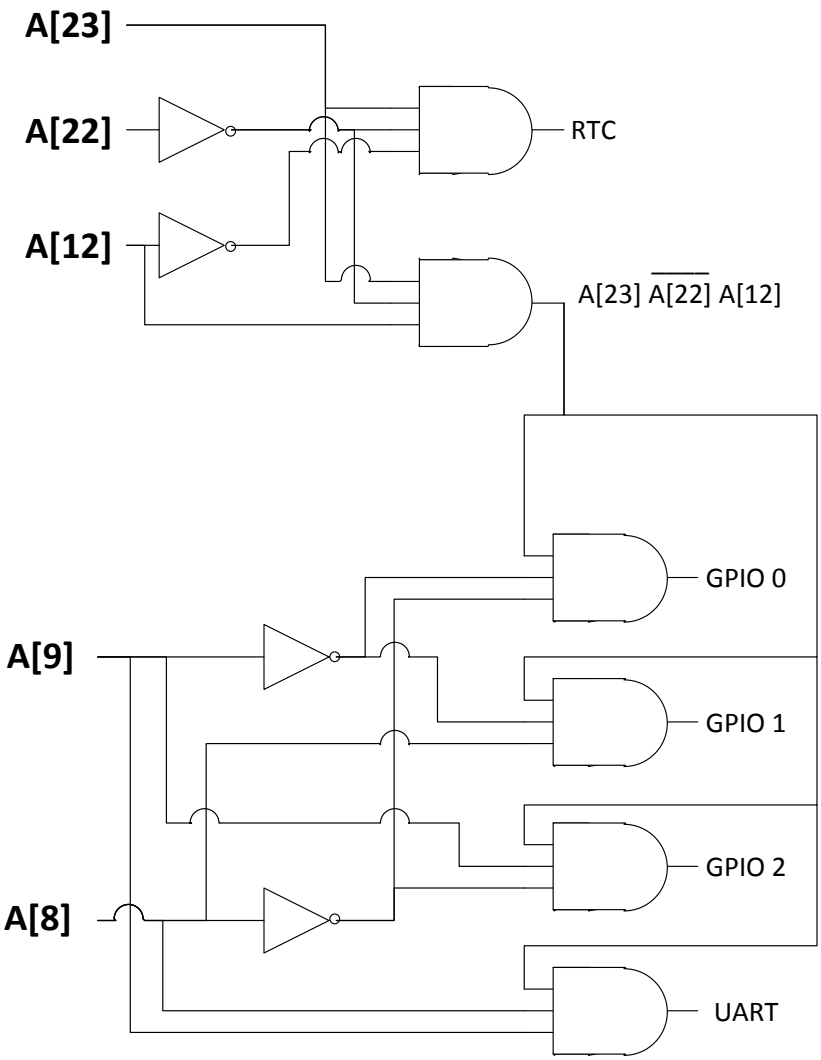


Figure 5.5, page 5-8, ARM Ltd., *AMBA Specification*, 2nd Revision, ARM 1999

Selection	A23	A22	A12	A9	A8	PSEL1: RTC	PSEL2: GPIO0	PSEL3: GPIO1	PSEL4: GPIO2	PSEL5: UART
RTC	1	0	0	-	-	1	0	0	0	0
GPIO0	1	0	1	0	0	0	1	0	0	0
GPIO1	1	0	1	0	1	0	0	1	0	0
GPIO2	1	0	1	1	0	0	0	0	1	0
UART	1	0	1	1	1	0	0	0	0	1


A[23]
A[22]
A[12]
RTC
A[23] $\overline{A[22]}$ A[12]
GPIO 0
A[9]
GPIO 1
GPIO 2
A[8]
UART

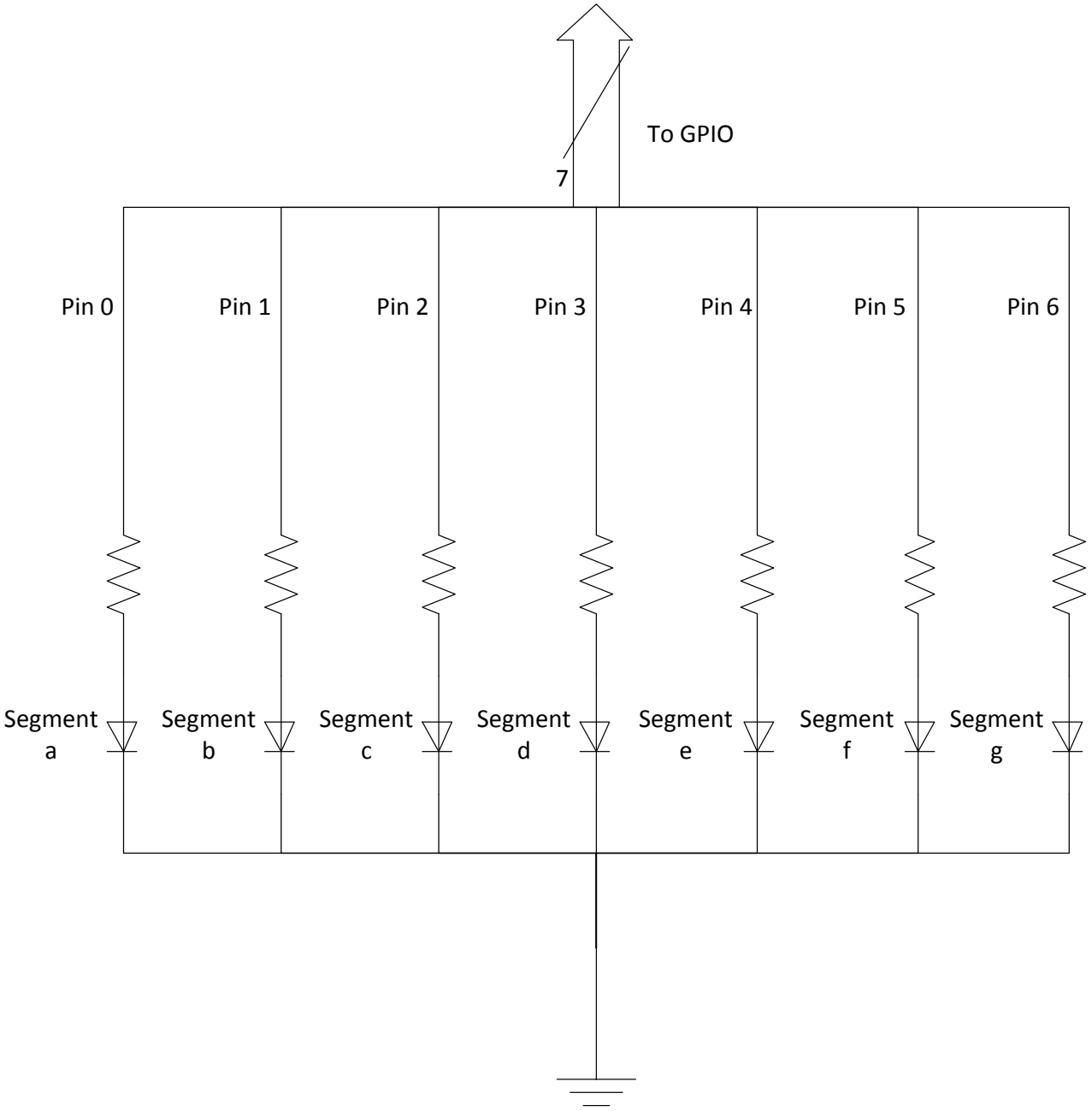
GPIO Design

The following components require the use of GPIO units. The total number of GPIO pins required for each is displayed in parentheses:

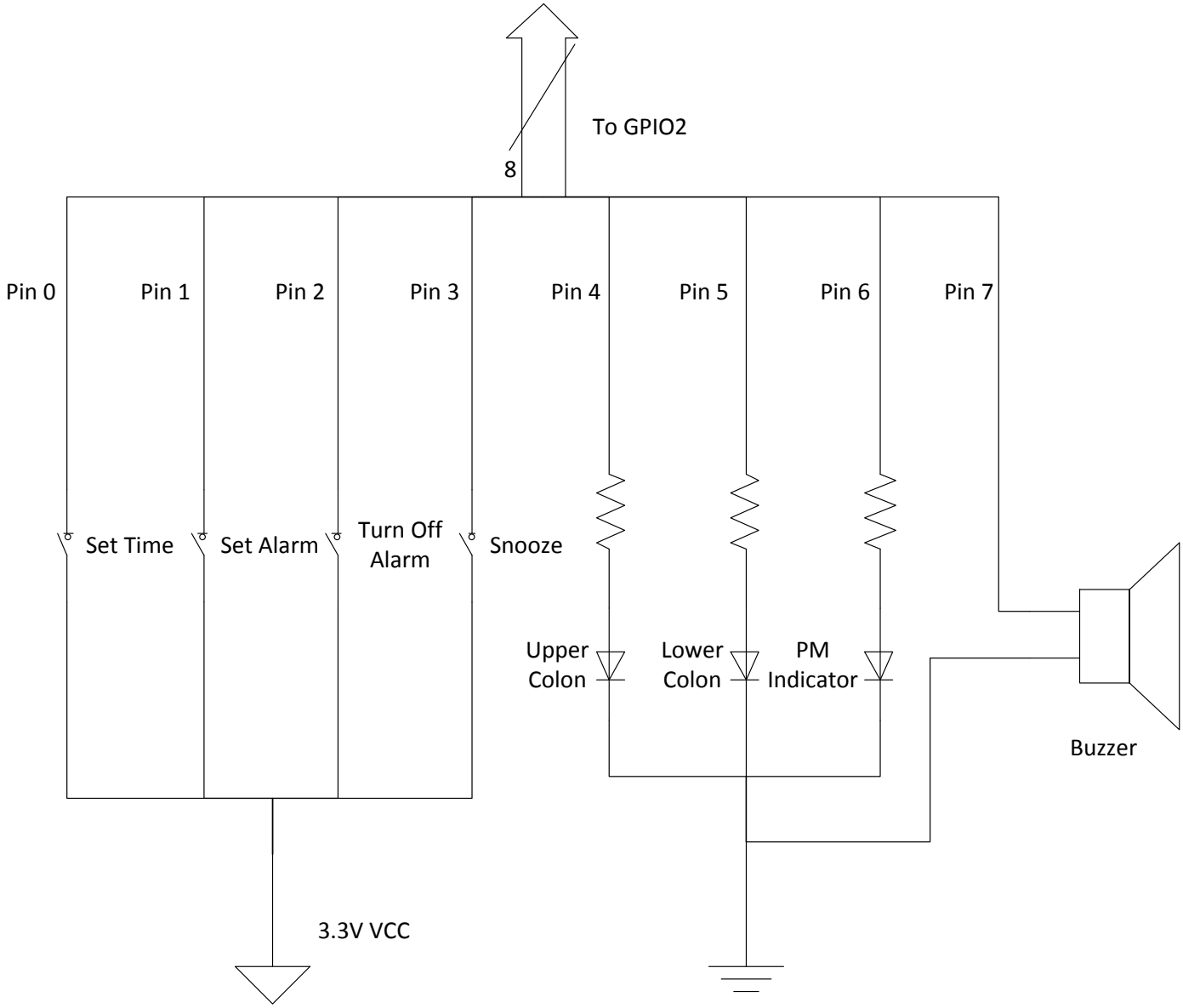
- 4 Seven-Segment Displays (28)
- 3 LEDs (3)
- 4 Momentary Push Buttons (4)
- 1 Buzzer (1)

In total, 36 GPIO pins are needed. Since each GPIO unit can support up to 16 pins (8 pins on Port A and 8 pins on Port B), three units will be required. The components are organized across the GPIO units as follows:

- GPIO0
  - Port A
    - Most significant hour digit (Seven-Segment Display) – 7 pins
  - Port B
    - Least significant hour digit (Seven-Segment Display) – 7 pins
- GPIO1
  - Port A
    - Most significant minute digit (Seven-Segment Display) – 7 pins
  - Port B
    - Least significant minute digit (Seven-Segment Display) – 7 pins
- GPIO2
  - Port A
    - 3 LEDs – 3 pins
    - 4 Momentary Push Buttons – 4 pins
    - 1 Buzzer – 1 Pin
  - Port B
    - Unused



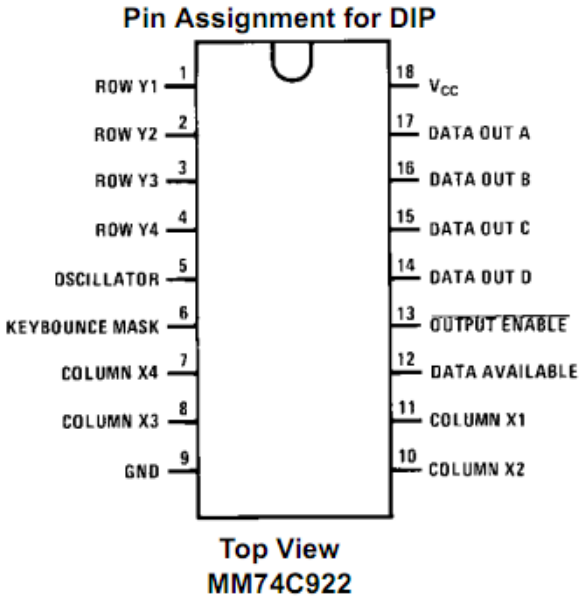
Here is the circuit diagram for a seven-segment display with respect to its connection to GPIO. It is designed as a common cathode, active high display. Note that with the selected seven-segment display the common cathode pins would connect to the ground bus of this diagram, while the lettered segment pins would connect to the GPIO pins listed above.



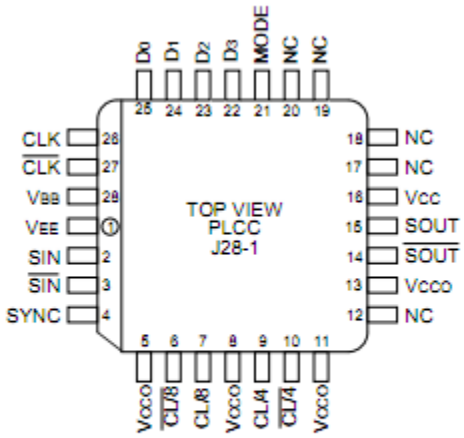
This circuit diagram shows the components attached to GPIO2. The bush buttons, LEDs, and buzzer are all designed to be active high components.

UART and Keypad

Since the keypad selected uses a matrix circuit, an encoding chip is used to get meaningful data from the keypad before it is connected to the UART. Below is a pin diagram for the encoder chip from its datasheet:



Since the keypad in use has only 3 columns rather than 4, pin 7 will simply be grounded. The rest of the row and column pins are connected to the corresponding outputs on the keypad. This will give us parallel data on pins 14 through 17. To finally connect this to the UART, a parallel to serial converter must be used. Shown below is the pin diagram from the selected converter's datasheet:

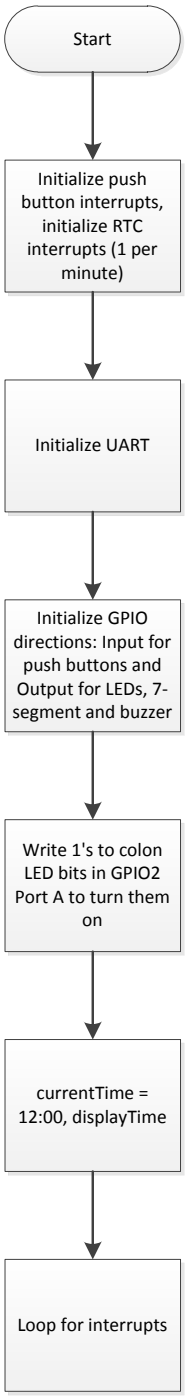


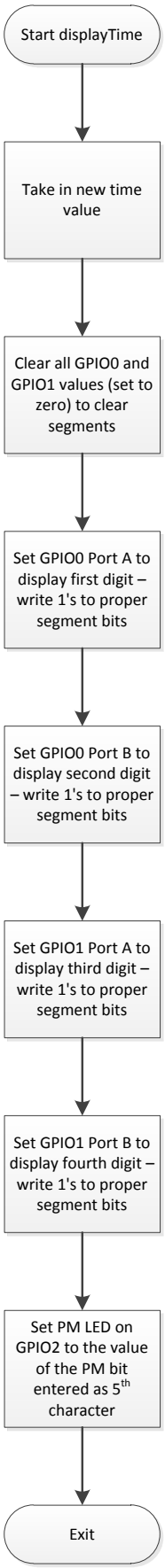
28-Pin PLCC (J28-1)

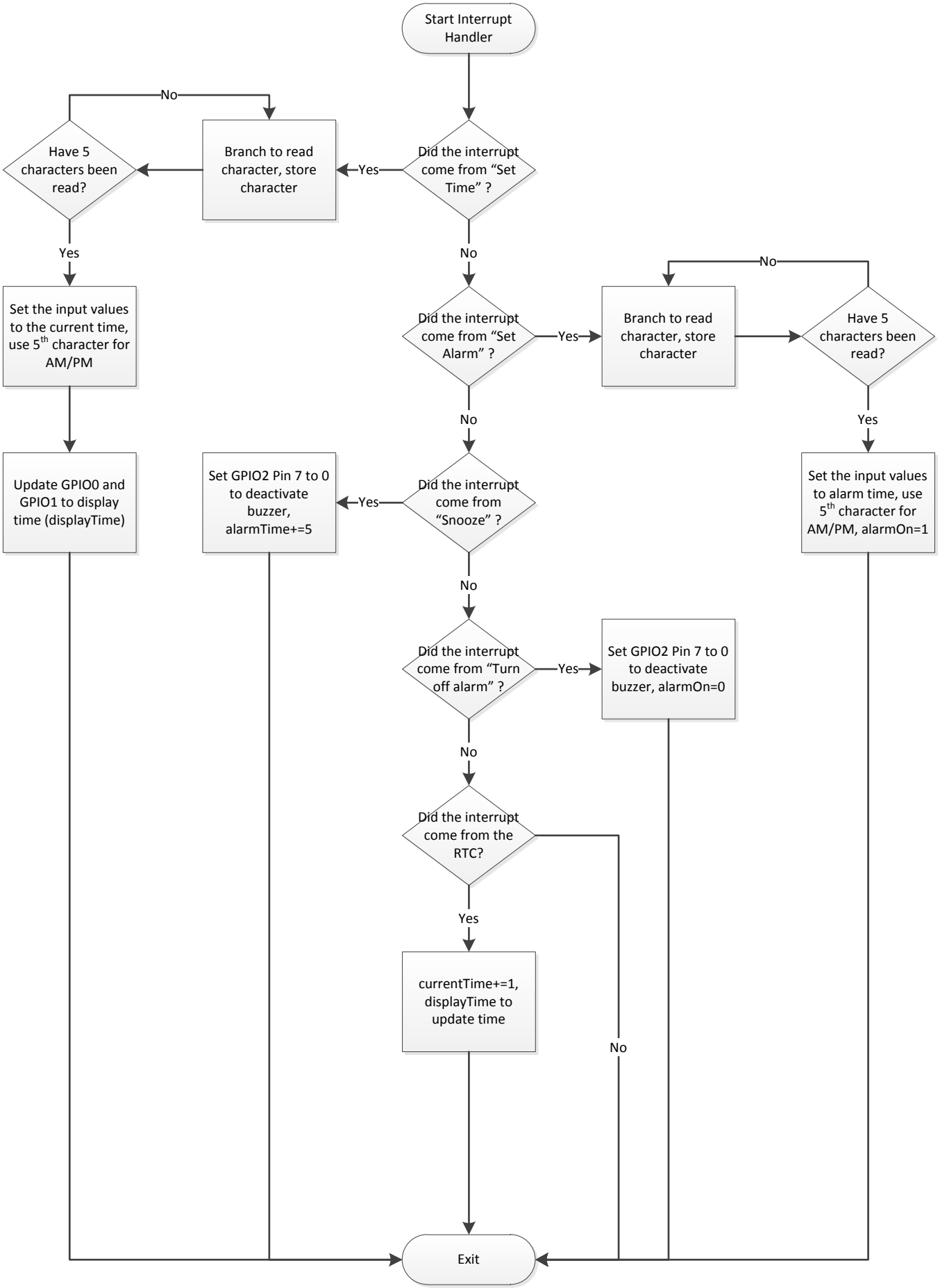
Here, pins 22 through 25 connect to the data out pins 14 through 17 on the MM74C922 encoder chip. Now, serial data out is finally achieved on pin 15, Serial Out, for connection to the UART.

Programming Model

For a working design, code needs to be designed as per the following flowcharts:






Start Interrupt Handler
Did the interrupt come from "Set Time" ?
Yes
Branch to read character, store character
Have 5 characters been read?
No
Yes
Set the input values to the current time, use 5th character for AM/PM
Update GPIO0 and GPIO1 to display time (displayTime)
No
Did the interrupt come from "Set Alarm" ?
Yes
Branch to read character, store character
Have 5 characters been read?
No
Yes
Set the input values to alarm time, use 5th character for AM/PM, alarmOn=1
No
Did the interrupt come from "Snooze" ?
Yes
Set GPIO2 Pin 7 to 0 to deactivate buzzer, alarmTime+=5
No
Did the interrupt come from "Turn off alarm" ?
Yes
Set GPIO2 Pin 7 to 0 to deactivate buzzer, alarmOn=0
No
Did the interrupt come from the RTC?
Yes
currentTime+=1, displayTime to update time
No
Exit

Resources

- Dr. Schindler's CSE379 Spring 2011 Lecture Notes
  - ARM Memory System
  - Advanced Microcontroller Bus Architecture (AMBA) Interface and General Purpose I/O
  - Universal Asynchronous Receiver/Transmitter (UART)
- Dr. Sridhar's CSE379 Spring 2007 Lecture Notes
  - Lecture 10 – AMBA Interface/GPIO
- ARM Technical Reference
  - ARM PrimeCell UART (PL010) Technical Reference Manual
  - ARM PrimeCell General Purpose Input/Output (PL060) Technical Reference Manual
  - ARM PrimeCell Real Time Clock (PL031) Technical Reference Manual
  - ARM AMBA Specification Reference Manual
- Digikey
- Datasheets, as listed with parts in the introduction