DESIGN AND CONSTRUCTION OF A BODY MASS INDEX CALCULATOR.

BY

OZUMBA CHUKWUMA TOCHUKWU

Reg. No.: 2003/15463EE

A Thesis submitted to the Department of Electrical and Computer Engineering, Federal University of Technology, Minna.

November 2008

DEDICATION

I dedicate this project to God Almighty who kept and sustained me throughout my study years and who is the source of my inspiration.

I would also love to dedicate this noble work to all security personnel and drivers that ever served during my study in this Great University.

DECLARATION

I Ozumba Chukwuma Tochukwu, declare that this work was done by me and has never been presented elsewhere for the award of a degree. I also hereby relinquish the copyright to the Federal University of Technology, Minna.

Ozumba Chukwuma T.	Mr. S. Oyewobi
(Student)	(Supervisor)
[illegible] 10/11/08	[illegible] 10/11/08
(Signature and Date)	(Signature and Date)
Dr. Y. Adediran	....
(H.O.D)	(External Examiner)
....	....
(Signature and Date)	(Signature and Date)

ACKNOWLEDGEMENT

I want to express my hearth felt gratitude to my parents Elder Bennett Ozumba and Mrs. Mercy Ozumba for their moral and financial support. They taught me the meaning of hard work and never stopped challenging me to bring out the best in me.

My regards also goes out to my supervisor Mr. Steven Oyewobi for his assistance and support and also for seeking to draw out the creativity in me. My thanks also goes to every single lecturer that once taught me during the course of my study, for their tremendous support and invaluable knowledge transfer for which I remain forever grateful.

Finally, I wish to thank all my friends and colleagues. I will forever cherish the moments I shared with you all. THANKS!

ABSTRACT

In this project a microcontroller. keypad. visual display unit and a battery were connected together in an appropriate manner to produce a calculator that would only calculate the "Body Mass Index" of users.

The calculator uses a 4x4 matrix keypad for inputting data. Thirteen out of the sixteen buttons of the keypad are useful while the remaining three are not connected. Digits from 0 to 9 are available on the keypad. To enter user's weight or height. buttons W and H are used respectively while button Cal is used for calculation.

The microcontroller (AT89C51) was programmed to read inputs from the keypad process the data and then send the result to the three digit 7-segment display unit.

A 9volts DC battery was used to power the circuit while a three digit 7-segment display unit was used for outputting.

List of Figures

List of Tables

CHAPTER ONE

INTRODUCTION

Body mass index (BMI), or Quetelet index, is a statistical measurement which compares a person's weight and height. Though it does not actually measure the percentage of body fat, it is a useful tool to estimate a healthy body weight based on how tall a person is. Due to it's ease of measurement and calculation, it is the most widely used diagnostic tool to identify obesity problems within a population. However it is not considered appropriate to use as a final indication for diagnosing individuals.[1]

It was invented between 1830 and 1850 by the Belgian polymath Adolphe Quetelet during the course of developing "social physics"[2]. Body mass index is defined as the individual's body weight divided by the square of their height. The formula universally used in medicine produce a unit of measure of kg/m^2 is used:

$$\text{BODY MASS INDEX} = \frac{\text{WEIGHT (kg)}}{\text{HEIGHT}^2\ (m^2)} \qquad \text{.........} 1.1$$

As a measure, BMI became popular during the early 1950s and 60s as obesity started to become a discernible issue in prosperous Western societies. BMI provided a simple numeric measure of a person's "fatness" or "thinness", allowing health professionals to discuss over and under-weight problems more objectively with their patients. However, BMI has become controversial because many people, including physicians, have come to rely on its apparent numerical authority for medical diagnosis, but that was never the BMI's purpose. It is meant to be used as a simple means of classifying sedentary (physically inactive) individuals with an average body composition.[3] For these individuals, the current value settings are as follows: a BMI of 18.5 to 25 may indicate optimal weight: a BMI lower than 18.5 suggests the person is underweight while a number above 25 may indicate the person is overweight; a BMI below 17.5 may indicate the person has anorexia or a related disorder; a number above 30 suggests the person is obese (over 40, morbidly obese). For a fixed body shape and body density, and given height, BMI is proportional to weight. However, for a fixed body shape and body density, and given weight, BMI is inversely proportional to the square of the height. So, if all body dimensions double, and weight scales naturally with the cube of the height, then BMI doubles instead of remaining the same. This result's in taller people having a reported BMI that is uncharacteristically high compared to their actual body fat levels. This anomaly is partially offset by the fact that many taller people are not just "scaled up" short people, but tend to have narrower frames in proportion to their height. It has been suggested that instead of squaring the body height (as the BMI does) or cubing the body height (as seems natural), it would be more appropriate to use an exponent of between 2.3 to 2.7.[4]

1.1 OBJECTIVES OF PROJECT

The main objectives of this project are as follow:

1. Stimulate the urge into more research into Body Mass Index (BMI) calculation using electronic components.
2. To design for the first time, a single device that calculates Body Mass Index.
3. To build a device that make's it easier for health workers to calculate the BMI of patients.
4. Make the point that the technology of microcomputer benefits all professions.

5. To prove that, an 8-bit microcontroller could be programmed to perform complex mathematical operations.

6. To provide the opportunity for wider knowledge in assembly language.

1.2 METHODOLOGY

The project is carried out by experimental modular design. The circuit is divided into blocks/modules and each module is analyzed extensively before its construction and tested after its construction so as to prove satisfactorily before joining the blocks to form the circuit. The blocks considered in this project are:

1. The power supply module.
2. The keypad unit.
3. The control unit (microcontroller).
4. The output (i.e LEDs).
5. The programming unit (source code generation).

1.3 BRIEF DESCRIPTION OF PROJECT

This project entails the use of a microcontroller as a base that would implement the mathematical operation. The keypad sends inputs (i.e weight and height) from the user. Base on the instructions in the microcontroller, these inputs (digits) are simultaneously displayed on the output screen. The microcontroller carries out the mathematical manipulation and then sends an output (BMI) to the seven segment LED display screen. When the first input (weight) is to be entered the "W" key on the keypad is pressed. While for the second input (height) the "H" key is pressed.

The block diagram of the BMI calculator is shown in Figure 1.0 below:

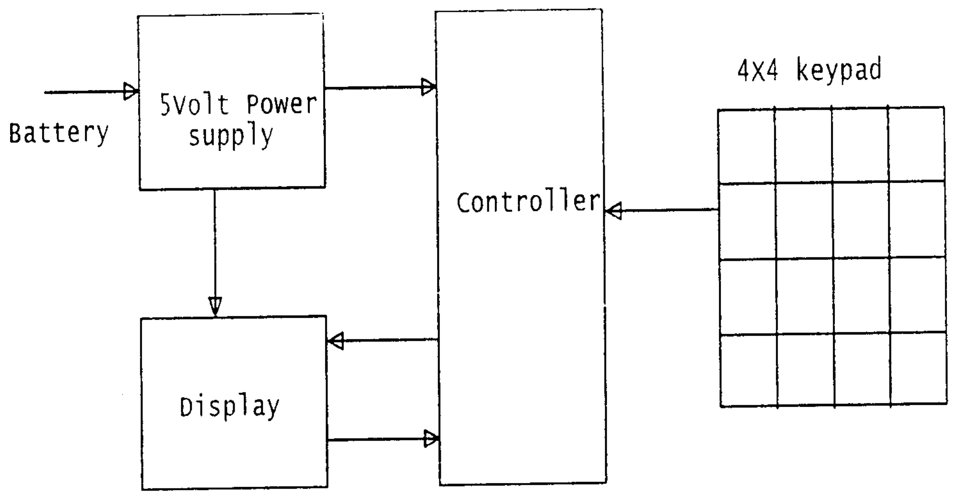

Fig.1.1 Block diagram of the BMI calculator.

1.4 PROJECT OUTLINE

This project seeks to design a net and portable device which can calculate Body Mass Index of a person and provide an effective and efficient means of archiving it. To fulfill this objective, this project report has been divided into chapters as follows:

Chapter one deals with the statement of the problem, the aim of the project, mode of operation, the methodology involved, the block diagram representation of the circuit all of which form the introduction.

Chapter two contains the literature review which gives the history of calculator, usage, categories and other related materials used in the course of the project.

The circuit design and analysis is dealt with in chapter three.

Chapter four has to do with construction and testing, the results obtained and some of the difficulties encountered.

Finally, chapter five caps it all up with the conclusion, recommendation, appendix and the reference materials consulted.

1.5 OTHER PERSONS INVOLVED.

Other contributors involved in this project work include professionals in the field of electrical and electronic engineering, computer engineering both within and outside the university community, technicians, fellow course mates and most especially my project supervisor.

CHAPTER TWO

LITERATURE REVIEW

A calculator is an electronic device for performing mathematical calculations. distinguished from a computer by a limited problem solving ability and an interface optimized for interactive calculation rather than programming. Calculators can be hardware or software. and mechanical or electronic. and are often built into devices such as PDAs or mobile phones.

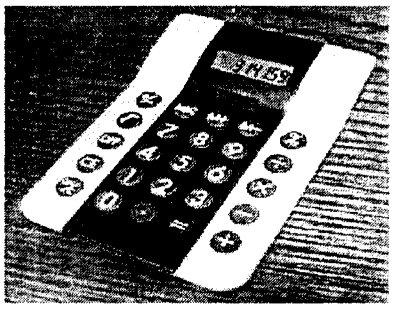

Fig.2.1 A basic calculator

Modern electronic calculators are generally small. digital. (often pocket-sized) and usually inexpensive. In addition to general purpose calculators. there are those designed for specific markets; for example. there are scientific calculators which focus on advanced mathematics like trigonometry and statistics. or even have the ability to do computer algebra. Modern calculators are more portable than most computers. though most PDAs are comparable in size to handheld calculators.

2.1 ORIGIN OF CALCULATOR: ABACUS

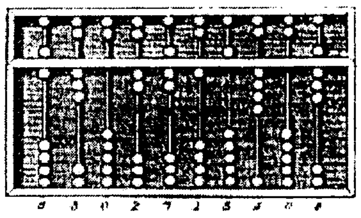

Fig.2.2 Chinese abacus.

The first calculators were abaci, and were often constructed as a wooden frame with beads sliding on wires. Abacuses were in use centuries before the adoption of the written Arabic numerals system and are still used by some merchants, fishermen and clerks in China and elsewhere.

2.2 OTHER EARLY CALCULATORS

Devices have been used to aid computation for thousands of years, using one-to-one correspondence with our fingers.[5] The earliest counting device was probably a form of tally stick. Later record keeping aids throughout the Fertile Crescent included clay shapes, which represented counts of items, probably livestock or grains, sealed in containers.[6]The abacus was used for arithmetic tasks. The Roman abacus was used in Babylonia as early as 2400 BC. Since then, many other forms of reckoning boards or tables have been invented. In a medieval counting house, a checkered cloth would be placed on a table, and markers moved around on it according to certain rules, as an aid to calculating sums of money (this is the origin of "Exchequer" as a term for a nation's treasury).

A number of analog computers were constructed in ancient and medieval times to perform astronomical calculations. These include the Antikythera Mechanism and the astrolabe from ancient Greece (c. 150-100 BC), which are generally regarded as the first mechanical analog computers.[7] Other early versions of mechanical devices used to perform some type of calculations include the planisphere and other mechanical computing devices invented by Abū Rayhān al-Bīrūnī (c. AD 1000); the equatorium and universal latitude-independent astrolabe by Abū Ishāq Ibrāhīm al-Zarqālī (c. AD 1015); the astronomical analog computers of other medieval Muslim astronomers and engineers; and the astronomical clock tower of Su Song (c. AD 1090) during the Song Dynasty.

Scottish mathematician and physicist John Napier noted multiplication and division of numbers could be performed by addition and subtraction, respectively, of logarithms of those numbers. While producing the first logarithmic tables Napier needed to perform many multiplications, and it was at this point that he designed Napier's bones, an abacus-like device used for multiplication and division.[8] Since real numbers can be represented as distances or intervals on a line, the slide rule was invented in the 1620s to allow multiplication and division operations to be carried out significantly faster than was previously possible.[9] Slide rules were used by generations of engineers and other mathematically inclined professional workers, until the invention of the pocket calculator. The engineers in the Apollo program to send a man to the moon made many of their calculations on slide rules, which were accurate to three or four significant figures.[10]

German polymath Wilhelm Schickard built the first digital mechanical calculator in 1623, and thus became the father of the computing era.[11] Since his calculator used techniques such as cogs and gears first developed for clocks, it was also called a 'calculating clock'. It was put to practical use by his friend Johannes Kepler, who revolutionized astronomy when he condensed decades of astronomical observations into algebraic expressions. An original calculator by Pascal (1640) is preserved in the Zwinger Museum. Machines by Blaise Pascal (the Pascaline, 1642) and Gottfried

Wilhelm von Leibniz (1671) followed. Leibniz once said "It is unworthy of excellent men to lose hours like slaves in the labour of calculation which could safely be relegated to anyone else if machines were used."[12]

2.3 THE DEVELOPMENT OF ELECTRONIC CALCULATORS

The first main-frame computers, using firstly vacuum tubes and later transistors in the logic circuits, appeared in the late 1940s and 1950s. This technology was to provide a stepping stone to the development of electronic calculators.

In 1954, IBM, in the U.S.A., demonstrated a large all-transistor calculator and, in 1957, the company released the first commercial all-transistor calculator, the IBM 608, though it was housed in several cabinets and cost about $80,000.[13] The Casio Computer Co., in Japan, released the Model 14-A calculator in 1957, which was the world's first all-electric "compact" calculator. It did not use electronic logic but was based on relay technology, and was built into a desk.

In October 1961, the world's first all-electronic desktop calculator, the Bell Punch/Sumlock Comptometer ANITA (A New Inspiration To Arithmetic/Accounting) was announced.[16][17] This British designed-and-built machine used vacuum tubes, cold-cathode tubes and Dekatrons in its circuits, with 12 cold-cathode "Nixie"-type tubes for its display. Two models were displayed, The Mk VII for continental Europe and the Mk VIII for Britain and the rest of the world, both for delivery from early 1962. The Mk VII was a slightly earlier design with a more complicated mode of multiplication and was soon dropped in favour of the simpler Mark VIII version. The ANITA had a full keyboard, similar to mechanical Comptometers of the time, a feature that was unique to it and the later Sharp CS-10A among electronic calculators. Bell Punch had been producing key-driven mechanical calculators of the Comptometer type under the names "Plus" and "Sumlock", and had realized in the mid-1950s that the future of calculators lay in electronics. They employed the young graduate Norbert Kitz, who had worked on the early British Pilot ACE computer project, to lead the development. The ANITA sold well since it was the only electronic desktop calculator available, and was silent and quick.

The tube technology of the ANITA was superseded in June 1963, by the U.S. manufactured Friden EC-130, which had an all-transistor design, 13-digit capacity on a 5-inch CRT, and introduced reverse Polish notation (RPN) to the calculator market for a price of $2200, which was about triple the cost of an electromechanical calculator of the time. Like Bell Punch, Friden was a manufacturer of mechanical calculators that had decided that the future lay in electronics. In 1964 more all-transistor elctronic calculators were introduced: Sharp introduced the CS-10A, which weighed 25 kg (55 lb) and cost 500,000 yen (~US$2500), and Industria Macchine Elettroniche of Italy introduced the IME 84, to which several extra keyboard and display units could be connected so that several people could make use of it (but apparently not at the same time).

There followed a series of electronic calculator models from these and other manufacturers, including Canon, Mathatronics, Olivetti, SCM (Smith-Corona-Marchant), Sony, Toshiba, and Wang. The early calculators used hundreds of Germanium transistors, since these were then cheaper than Silicon transistors, on multiple circuit boards. Display types used were CRT, cold-cathode Nixie tubes, and filament lamps. Memory technology was usually based on the delay line memory or the magnetic core memory, though the Toshiba "Toscal" BC-1411 appears to use an early form of dynamic RAM built from discrete components. Already there was a desire for smaller and less power-hungry machines.

The Olivetti Programma 101 was introduced in late 1965; it was a stored program machine which could read and write magnetic cards and displayed results on its built-in printer. Memory, implemented by an acoustic delay line, could be partitioned between program steps, constants, and data registers. Programming allowed conditional testing and programs could also be overlaid by reading from magnetic cards. It is regarded as the first personal computer produced by a company (that is, a desktop electronic calculating machine programmable by non-specialists for personal use). The Olivetti Programma 101 won many industrial design awards.

The Monroe Epic programmable calculator came on the market in 1967. A large, printing, desk-top unit, with an attached floor-standing logic tower, it was capable of being programmed to perform many computer-like functions. However, the only branch instruction was an implied unconditional branch (GOTO) at the end of the operation stack, returning the program to its starting instruction. Thus, it was not possible to include any conditional branch (IF-THEN-ELSE) logic. During this era, the absence of the conditional branch was sometimes used to distinguish a programmable calculator from a computer. The first handheld calculator was developed by Texas Instruments in 1967. It could add, multiply, subtract, and divide, and its output device was a paper tape.[14]

2.4 TECHNICAL IMPROVEMENTS

Through the 1970s the hand-held electronic calculator underwent rapid development. The red LED and blue/green vacuum-fluorescent displays consumed a lot of power and the calculators either had a short battery life (often measured in hours, so rechargeable Nickel-Cadmium batteries were common) or were large so that they could take larger, higher capacity batteries. In the early 1970s Liquid crystal displays (LCDs) were in their infancy and there was a great deal of concern that they only had a short operating lifetime. Busicom introduced the Busicom LE-120A "HANDY" calculator, the first pocket-sized calculator and the first with an LED display, and announced the Busicom LC with LCD display. However, there were problems with this display and the calculator never went on sale. The first successful calculators with LCDs were manufactured by Rockwell International and sold from 1972 by other companies under such names as: Dataking LC-800, Harden DT/12, Ibico 086, Lloyds 40, Lloyds 100, Prismatic 500 (a.k.a P500), Rapid Data Rapidman 1208LC. The LCDs were an early form with the numbers appearing as silver against a dark background. To present a high-contrast display these models illuminated the LCD using a filament lamp and solid plastic light guide, which negated the low power consumption of the display. These models appear to have been sold only for a year or two.

A more successful series of calculators using the reflective LCD display was launched in 1972 by Sharp Inc with the Sharp EL-805, which was a slim pocket calculator. This, and another few similar models, used Sharp's "COS" (Crystal on Substrate) technology. This used a glass-like circuit board which was also an integral part of the LCD. In operation the user looked through this "circuit board" at the numbers being displayed. The "COS" technology may have been too expensive since it was only used in a few models before Sharp reverted to conventional circuit boards, though all the models with the reflective LCD displays are often referred to as "COS".

In the mid-1970s the first calculators appeared with the now "normal" LCDs with dark numerals against a grey background, though the early ones often had a yellow filter over them to cut out damaging UV rays. The big advantage of the LCD is that it is passive and reflects light, which requires much less power than generating light. This led the way to the first credit-card-sized calculators, such as the Casio Mini Card LC-78 of 1978, which could run for months of normal use on a couple of button cells.

There were also improvements to the electronics inside the calculators. All of the logic functions of a calculator had been squeezed into the first "Calculator on a chip" integrated circuits in 1971, but this was leading edge technology of the time and yields were low and costs were high. Many calculators continued to use two or more integrated circuits (ICs), especially the scientific and the programmable ones, into the late 1970s. The power consumption of the integrated circuits was also reduced, especially with the introduction of CMOS technology. Appearing in the Sharp "EL-801" in 1972, the transistors in the logic cells of CMOS ICs only used any appreciable power when they changed state. The LED and VFD displays had often required additional driver transistors or ICs, whereas the LCD displays were more amenable to being driven directly by the calculator IC itself. With this low power consumption came the possibility of using solar cells as the power source, realized around 1978 by such calculators as the Royal Solar 1, Sharp EL-8026, and Teal Photon.

CHAPTER THREE

CIRCUIT DESIGN AND ANALYSIS

In order to realize the design of this project. the project was divided into functional units or sub-circuits. The coupling together of these functional units resulted in the final circuit. The Body Mass Index (BMI) calculator was designed around the under-listed subsystems:

i. Power supply.

ii. 4 x 4 matrix keypad.

iii. 8-bit controller.

iv. Three digit 7-segment display.

v. The system ware/ firmware

3.1 POWER SUPPLY

For portability, a battery derived power source was selected for system operational power. A 9volts dc battery was chosen as the primary system power source.

Since the logic subsystem requires a regulated 5volts. a 5volts regulator was interpolated between the 9volts supply and the logic subsystem as shown in the diagram below.

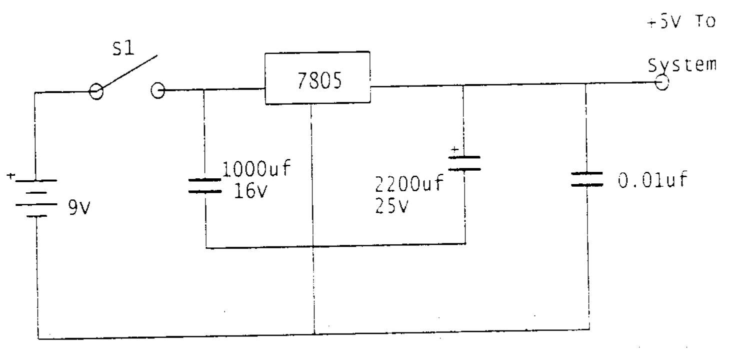

Fig.3.1 5volts power source

Input current required was calculated to know the required regulator from the 78xxseries:

Table 3.1 the system current requirement

SYSTEM DEVICE	CURRENT DRAIN
Microcontroller	15Ma
7-Segment display	240Ma
Total current drain	255Ma

Since the supply current drain is less than 1Amp, a 1Amp 5volts regulator was selected.

The 9volts DC source was buffer by a 16volts 1000uF capacitance. The 5volts output was held stabilized by a second 2200uF capacitance as shown in Fig. 3.0 above. A 0.01uF capacitor was place across the low- voltage dc output for high frequency noise filtering.

3.2 FOUR BY FOUR MATRIX KEYPAD

Keypad was required for data entry. To save on port pins, a 4 x 4 matrix arrangement was used. The keypad was converted to port 3 of the AT8951 system controller as shown in fig. 3.2 below:

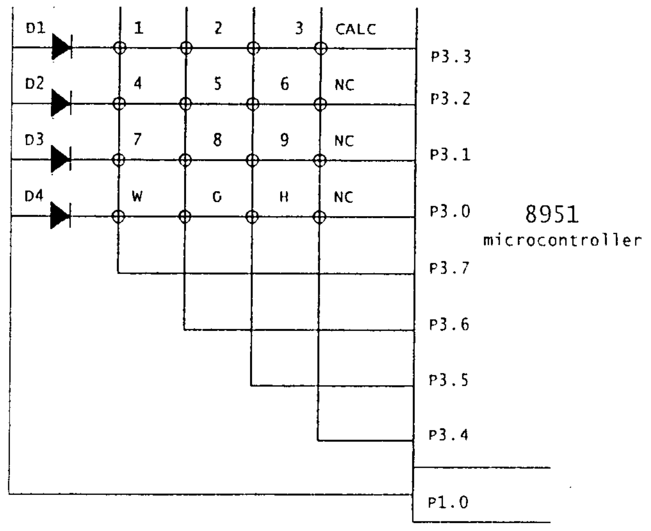

Fig 3.2 A 4*4 Matrix to System Microcontroller

Keypad scanning was carried-out by placing a logical "0" on the upper nibble of port 3, while the lower nibble was set to logic "1".

A 4-diode arrangement enables key presses to be detected as illustrated below: Assuming key #1 is pressed, Port 3 pin 3 goes low due to the interconnect of an already logic "0" in port 3 pin 7. Diode D1 is forward biased and conducts pulling Port1 pin0 low. The software loop on this port pin (P1.0) to detect key closure. Consequently, upon Port1 pin0 going low, the scan key routine is executed. The low order nibble of port 3 is scanned in sequence (i.e P3.3, P3.2, P3.1, P3.0). If a zero is detected on Port 3 pin3, a "0" is stored in a RAM variable key. If Port 3 pin2 is low, a "4" is stored, for Port 3 pin1 a "7" while for Port 3 pin0 an "11" would be stored.

The port pins are then flipped over, that is, the higher nibble made zero. The upper nibble is then scanned sequentially. If a low is detected on port 3 pin7, a value "1" is stored in a second RAM variable. But if port 3 pin6 is low a "2" is stored, while if a low is detected on port 3 pin5 a "3" stored, also for a low port 3 pin4 a "4" is stored.

The digit value deduced from previous scan is added to the second scan. The value is loaded into the accumulator and used to reference a look-up table. Thirteen keys out of the sixteen keys available in the keypad were used, keys 0-9, weight key, height key and calculate key. The three unassigned keys were discarded in software even when pressed.

3.3 EIGHT-BIT SYSTEM CONTROLLER

A programmable logic device (AT8951) was implemented as the system controller. The device was chosen for it's flexible hardware design. The device has the specifications as stated on the datasheet.[15]

The controller was programmed to perform the following:

i. Read input data from keypad.

ii. Process the data.

iii. Display result on the 7-segment display.

The controller was interfaced with the keypad over port 3 and the display port over port 0 and the digit driver over port 2.

3.4 THREE DIGIT 7-SEGMENT VISUAL DISPLAY UNIT

A three digit 7-segment VDU was incorporated into the design for data display. The display was multiplexed as shown below to save on the number of port pins needed for a multi-digit implementation.

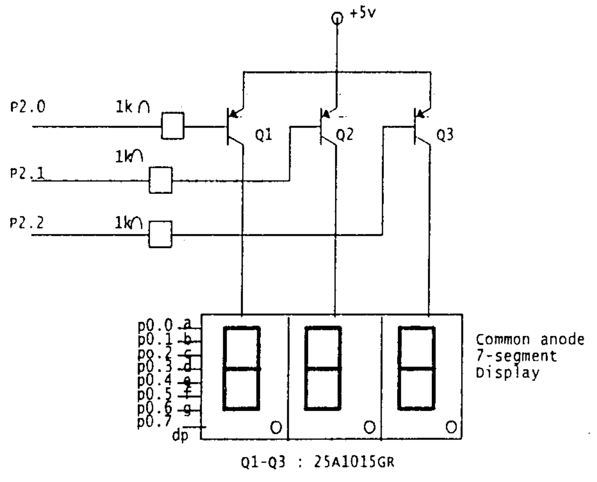

Fig.3.3 A 3-digit display

The digits shared a common data path, Port 0. Individual control over digit positioi was implemented via three PNP anode drivers as show above in fig.3.2. In a multiplexed display, time-slice allocation of the common data path is utilized. In this implementation, the data to be displayed are present on the common data path for a slice of time. During this time allocation, the corresponding digit driver is turned on to source current through the segments. Common-anode displays were used, hence PNP drivers were needed to switch current through the LEDs. The Port 0 providing a return path for the lit iyu display segments. A typical control loop for display ie data on a multiplexed unit is given below:

i. Turn off all digit drivers.

ii. Place data to show on common port.

iii. Turn on digit driver associated with the display position.

iv. Delay for persistence.

v. Turn off digit driver.

vi. Repeat steps i-v, changing the controlled digit driver.

If the above loop is executed at least about 50 times per second, there is the illusion of a continuously-on individual digit display.

3.5 DESIGN CALCULATIONS

For an n-digit display, the current through each segment is n multiply by segment continuous forward-current. This is to overcome the current discontinuity arising from the multiplexing effect. For a continuous current of 15mA, a three-digit display requires 45mA pulsed current. The current drawn by a fully on 8-segment display is 8*45mA =360mA. This value of current must be handled by each digit driver.

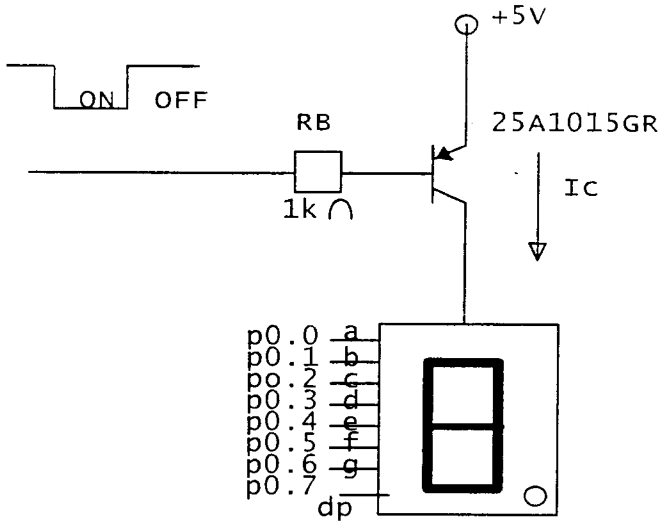

Fig.3.4 Single digit 7-segment display.

The 2SA1015GR digit driver has a gain of 200.

$$I_B = \frac{I_C}{H_{fe}} = \frac{360mA}{200} \qquad \text{......} 3.1$$

$$= 1.8mA$$

The value of base resistance was deduced from the expression:

$$R_B = \frac{V_{CC} - V_{BE}}{I_B} \qquad \text{......} 3.2$$

$$= \frac{5 - (0.7 X10\text{^}3)}{1.8}$$

$$=2400\text{ohms}$$

A 1000ohms resistance was used to allow for a fairly brighter display.

3.6 SYSTEM SOFTWARE

The system firmware was coded in assembly for speed. The software was modularized to aid easy debugging and fine tuning. The listing is included in the Appendix.

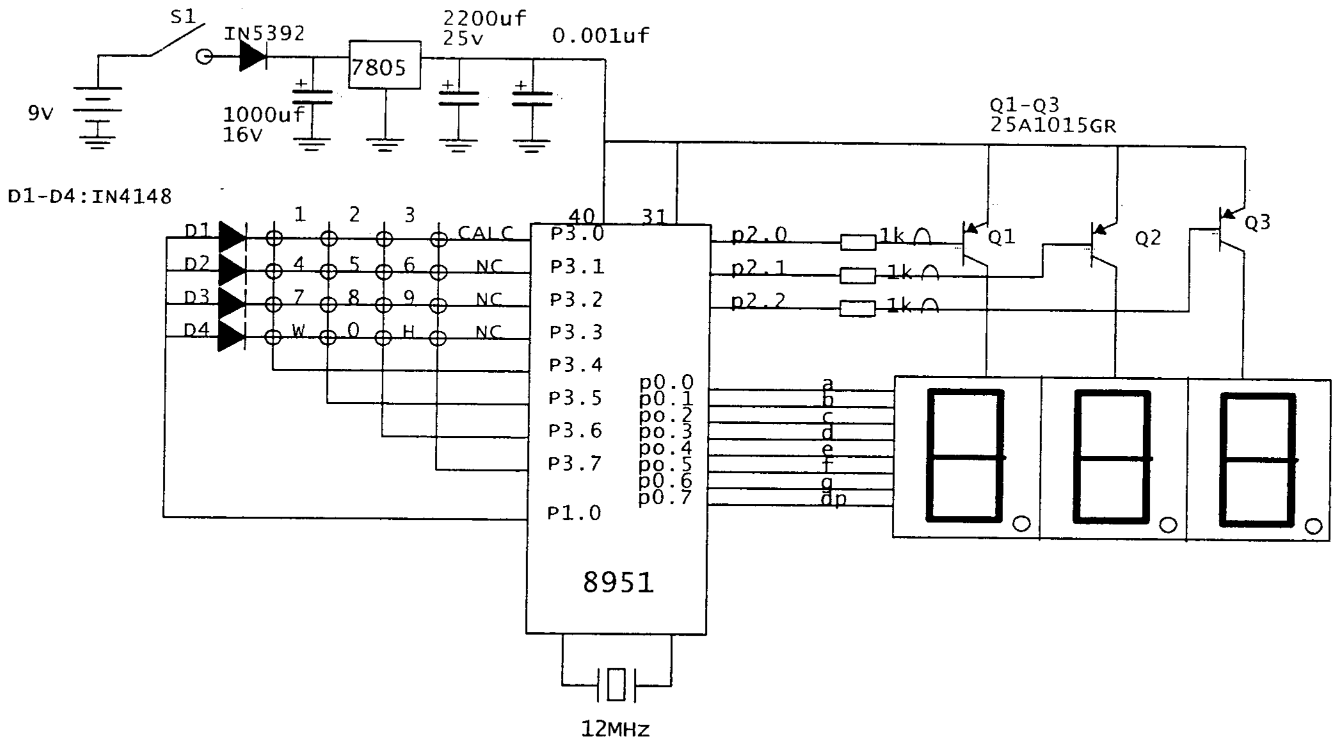

Fig.3.5 the complete circuit diagram.

CHAPTER FOUR

CONSTRUCTION AND TESTING

4.1 CONSTRUCTION

In order to facilitate construction, the project was first constructed on a bread board. This helps to facilitate easy troubleshooting of the circuit. Also this ensures that all faults detected are corrected before the components are finally soldered together on a printed circuit board (PCB). This is very important to prevent multiple soldering and de-soldering of circuit components from the printed circuit board (PCB) in the event of fault detection as well as protect the components from damage.

Care was exercised when handling components particularly the sensitive ones (such as ICs) to avoid damage due to static.

Components such as: diodes, capacitors and transistors were connected properly with respect to their polarities to avoid damage or mal-functioning. The components were well spaced during connection for easy access to faulty components in addition to providing ease in fault tracing. Proper spacing also helps to check capacitive build up between components when they are connected in close proximity. After soldering, the leads were trimmed with a sniper. Also during deflecting (de-soldering) a small brush was used to dislodge stubborn globes of flux.

4.2 TESTING/RESULTS

Possible BMI of ten (10) different users were calculated using both electronic calculator and the constructed BMI calculator. The results obtained are as shown in table 4.1 below:

Table 4.1 Results obtained

Test	Weight(kg)	Height(m)	BMI(kg/m2)*	BMI(kg/m2)**
1	90	1.50	40.0	40
2	100	1.60	39.0	39.0625
3	70	1.70	24.2	24.2214
4	60	1.80	18.5	18.5185
5	67	1.90	18.5	18.5596
6	70	2.00	17.5	17.5
7	100	2.10	22.6	22.6757
8	100	2.20	20.6	20.6612
9	120	2.50	19.2	19.2
10	260	1.50	000	115.5556

**from an electronic calculator

*from the constructed BMI calculator

4.3 DISCUSSION OF RESULT

Since the microcontroller used is an eight-bit controller, it cannot calculate for user with weight greater than 255kg and also users with body mass index greater than 999kg/m2. Although users with such conditions need not look for a machine to calculate their BMI since they are already at risk.

Comparing the results obtained from an electronic calculator with that of the constructed BMI calculator, some level of error was noticed, since the BMI calculator was designed to display the results to three significant figures.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATION

5.1 CONCLUSION

The project was carried out to make body mass index calculation easier and faster. The results obtained reveal the efficiency and accuracy of the project work as discussed in chapter four of the project report.

The project also reveals that an 8-bit microcontroller can be used to perform complex mathematical operations, like calculating body mass index.

5.2 RECOMMENDATION

Improvement can further be carried out on the project in the under listed aspects:

i. Change the output display from LED to LCD.

ii. Increase the output display to more than three significant figures.

iii. Make the device perform more functions other than just calculating BMI.

iv. Make the device more portable and fanciful.

v. Make the device more user-friendly

REFERENCES

[1] http://www.medicinet.com/script/main/art.asp?articlekey=56149$page=2

[2] Eknoyan Garabed, "Adolphe Quetelet (1796-1874)--the average man and indices of obesity". *Nephrol. Dial Transplant .vol* 23 No.1, January2008, pp.47–51.

[3] WHO Technical Report Series, #854, *Physical Status: The Use and Interpretation of Anthropometry*, Pg. 9

[4] http://www.math.utah.edu/~korevaar/ACCESS2003/bmi.pdf

[5] Georges Ifrah notes that humans learned to count on their hands. Ifrah shows, for example, a picture of Boethius (who lived 480–524 or 525) reckoning on his fingers in Ifrah 2000, pp. 48.

[6] http:///en.wikipedia.org/wiki/calculator#CITEREFSchmandt-Besserat1981

[7] http:/en.wikipedia.org/wiki/calculator#CITEREFLlazos1994

[8] Ancient Discoveries, Episode 11: Ancient Robots, History Channel 6 September 2008

[9] Kells, Kern & Bland 1943, pp. 92

[10] http://en.wikipedia.org/wiki/calculator#CITEREFSchmidhuber

[11] http://en.wikipedia.org/wiki/calculator#CITEREFSmith1929

[12] http://www-03.ibm.com/ibm/history/exhibits/vintage/vintage_4506w2214.html

[13] http://www.education.ti.com/educationportal/sites/us/ninproductsingle/about_press-release_news37.html

[14] http://www.npr.org/templates/story/story.php?storyId=14845433

[15] http://www.atmel.com/atmel/acrobat/doc0265.pdf

APPENDIX

```
INCLUDE 89c51.mc


dx_port EQU p2

dataport EQU p0

weight_0 DATA 08h

weight_1 DATA 09h

weight_2 DATA 0ah

height_0 DATA 0bh

height_1 DATA 0ch

height_2 DATA 0dh

result_0 DATA 0eh

result_1 DATA 0fh

result_2 DATA 10h

weight_hi DATA 11h

weight_lo DATA 12h

height_hi DATA 13h

height_lo DATA 14h

quotient_hi DATA 15h

quotient_lo DATA 16h

remainder_hi DATA 17h

remainder_lo DATA 18h

max_loc DATA 19h

data_0 DATA 1ah

data_1 DATA 1bh

data_2 DATA 1ch

data_3 DATA 1dh
```

```
dx_Ctrl DATA 1eh
count DATA 1fh
temp DATA 30h
temp1 DATA 31h
weight_hi DATA 32h
weight_lo DATA 33h
height_hi DATA 34h
height_lo DATA 35h
temp_hi DATA 36h
temp_lo DATA 37h
key_code DATA 38h
count2 DATA 39h
count3 DATA 3ah
count4 DATA 3bh
height_0_temp DATA 3ch
height_1_temp DATA 3dh
height_2_temp DATA 3eh
weight_0_temp DATA 3fh
weight_1_temp DATA 40h
weight_2_temp DATA 41h
dp_pos1 DATA 42h
dp_pos2 DATA 43h
weight_temp_hi DATA 44h
weight_temp_lo DATA 45h
weight_x DATA 46h
weight_y DATA 47h
dp_pos_1 BIT 127
dp_pos_2 BIT 126
```

```
weight_ptr BIT 125

height_ptr BIT 124

key_Valid BIT 123

show_Weight BIT 122

show_height BIT 121

show_Result BIT 120

weight_mask EQU 0ah

height_mask EQU 0bh

calculate_mask EQU 0fh

stack EQU 80

row_1 BIT p3.3

row_2 BIT p3.2

row_3 BIT p3.1

row_4 BIT p3.0

col_1 BIT p3.7

col_2 BIT p3.6

col_3 BIT p3.5

col_4 BIT p3.4

key_in BIT p1.0

weight_led BIT p2.4

height_led BIT p2.5

sounder_dx BIT p2.6




;***************************************************************

org 0000h

start_up:    CLR ea
```

```
                    MOV sp,#stac
            ACALL reset_delay
                    ACALL rese delay
                    ACALL reset_delay
                    ACALL reset_delay
                    MOV p3,#00001111
            SETB key_in
            CLR weight_ptr
            CLR height_ptr
            CLR show_Weight
            CLR show_height
            SETB show_result
            MOV R0,#height_0
loop:       MOV @R0,#0
            INC R0
            cjne r0,#height_0+10, loop
            MOV R0,#height_0_temp
loop_1:       MOV @R0,#0ffh
                    INC R0
                    CJNE R0,#height_0_temp+10, loop_1


mainloop:     ACALL write_display
                    JB key_in, mainloop
            ACALL scan_key
            JNB key_valid, mainloop
            CLR key_valid
            ACALL parse_key
            SJMP mainloop
```

```
;*************************************************************

 show_long:          MOV count2,#60

 long_loop:          ACALL write_now

                     DJNZ count2, long_loop

                     RET

;**************************************************************

parse_key:  CJNE A, #weight_mask, go_1

        ACALL reset_weight

        ACALL write_display

                     RET

;**********************************************************

go_1:     CJNE A, #height_mask, go_2

        ACALL reset_height

        ACALL write_display

        RET

;**************************************************************

go_2:     CJNE A, #calculate_mask, go_3

                     ACALL compute_bmi

        RET

;**************************************************************

go_3:     CJNE A, #9, go_4

skip_Back:  ACALL put_data

        ACALL write_display

        RET

;****************************************************************

go_4:     JC skip_Back

        ACALL write_display
        RET

;*****************************************************

put_data:  JB height_ptr, store_height

        JB weight_ptr, store_weight

        ACALL write_display

        RET

;*****************************************************

store_Height: MOV dp_pos1,#01111111b

                      MOV dp_pos2,#0ffh

                      MOV height_0, height_1

        MOV height_1, height_2

        MOV height_2, A

        MOV height_0_temp, height_1_temp

        MOV height_1_temp, height_2_temp

        MOV height_2_temp,A

        ACALL write_display

        RET

;*****************************************************

store_weight:  MOV weight_0, weight_1

                      MOV dp_pos1,#0ffh

                      MOV dp_pos2,#0ffh

        MOV weight_1, weight_2

        MOV weight_2, A

        MOV weight_0_temp,weight_1_temp

        MOV weight_1_temp, weight_2_temp

        MOV weight_2_temp, A

        ACALL write_display

        RET
```

```
;*****************************************************

reset_Weigh:  SETB height_led

        CLR weight_led

        CLR height_ptr

        SETB weight_ptr

        CLR A

        MOV weight_0,A

        MOV weight_1, A

        MOV weight_2, A

        MOV weight_0_temp,#0ffh

        MOV weight_1_temp,#0ffh

        MOV weight_2_temp,#0ffh

        CALL write_display

        RET

;*******************************************************

reset_height:  SETB weight_led

        CLR height_led

        SETB height_ptr

        CLR weight_ptr

        CLR A

        MOV height_0,A

        MOV height_1,A

        MOV height_2,A

        MOV height_0_temp,#0ffh

        MOV height_1_temp,#0ffh

        MOV height_2_temp,#0ffh

        ACALL write_display

        RET
```

```
;**************************************************************

compute_bmi:          MOV A, height_0

                 ORL A, height_1

                 ORL A, height_2

                 JZ zero_result

                 MOV A, weight_0

                 ORL A, weight_1

                 ORL A, weight_2

                 JZ zero_result

;**************************************************************

        ;new additions

                 MOV A, weight_0

                 ORL A, weight_1

                 ORL A, height_0

                 ORL A, height_1

                 JZ zero_result




                 MOV A,weight_0

                 ORL A, weight_1

                 ORL A, height_0

                 JZ zero_result





       ;additions

                 ;MOV A, height_0

                 ;ORL A, height_1
```

```
            ;JZ zero_result

            ;MOV A,weight_0

            ;ORL A, weight_1

            ;JZ zero_result

            ACALL go_now

;            ;ACALL chk_dp_pos2

            RET

;***************************************************************

zero_result:    SETB show_Result

                        CLR height_ptr

            CLR weight_ptr

            SETB weight_led

            SETB height_led

            MOV result_0,#0

            MOV result_1,#0

            MOV result_2,#0

            ACALL write_display

            RET

;***************************************************************

go_now:                 ACALL process_height

                             ACALL process_Weight

                             MOV A, weight_hi

                             JNZ zero_result

;new nsertions from line below:

;;;************************************************************

                             MOV A, weight_lo

                             CJNE A,#255, continue_now

                             MOV A,height_0
                        JNZ continue_now
                        MOV A, height_1
                        MOV B,#10
                        MUL ab
                        ADD A, height_2
                        CJNE A,#63, bbb_a
                        ;CJNE A,#51, BBB_A
                        SJMP continue_now



bbb_a:              JNC continue_now
                        SJMP zero_result


;*************************************************************
; nserton stops here:
continue_now:       MOV count,#99
                        ACALL mul_weight_100_10
                        ACALL find_bmi
                        ACALL hex2bcd
                        ACALL show_bmi
                        RET


;******************************************************
show_bmi:           MOV result_0,R7
                        MOV result_1, R6
                        MOV result_2, R5
                        SETB show_result
                        CLR show_weight
```

```
                                CLR show_height

                                CLR weight_ptr

                                CLR height_ptr

                                ACALL write_display

                                RET



                    ;
;************************************************************

process_Weight:     MOV A, weight_0

                          MOV B,#100

                          MUL ab

                          MOV weight_hi, B

                          MOV weight_lo, A

                          MOV A, weight_1

                          MOV B,#10

                          MUL ab

                          ADD A, weight_lo

                          MOV weight_lo, A

                          CLR A

                          ADDC A, weight_hi

                          MOV weight_hi, A

                          MOV A, weight_2

                          ADD A, weight_lo

                          MOV weight_lo, A

                          CLR A

                          ADDC A , weight_hi

                          MOV weight_hi, A

                          MOV weight_x, weight_hi             ; nuevo
                    MOV weight_y. weight_lo                ; nuevo

                    MOV weight_temp_hi,weight_hi

                    MOV weight_temp_lo. weight_lo

                    RET
;*****************;***************************************************************

process_height:     MOV A, height_0

                    MOV B,#100

                    MUL ab

                    MOV height_hi, B

                    MOV height_lo, A

                    MOV A, height_1

                    MOV B,#10

                    MUL ab

                    ADD A, height_lo

                    MOV height_lo, A

                    CLR A

                    ADDC A, height_hi

                    MOV height_hi, A

                    MOV A, height_2

                    ADD A, height_lo

                    MOV height_lo, A

                    CLR A

                    ADDC A , height_hi

                    MOV height_hi, A

                    RET


;*******************************************************************************

mul_weight_100_10:MOV temp, weight_lo
```

```
                        MOV temp1, weight_hi

mul_100_loop:      MOV A , temp

                        ADD A,weight_lo

                        MOV weight_lo, A

                        CLR A

                        ADDC A, weight_hi

                        ADDC A, temp1

                        MOV weight_hi, A

                        DJNZ count, mul_100_loop

                        RET


;*****************************************************************

;correct code version

find_bmi:          ACALL 16_bit_div

                        MOV remainder_hi, R1        ;store remainder here

                        MOV remainder_lo, R0        ;store remainder here

                        MOV weight_hi, R3           ;store quotinet_hi here

                        MOV weight_lo,R2            ; store_lo,a qutient_lo here

                        MOV count,#99               ; load for *100

                        ACALL mul_weight_100_10     ;multiply quotient by 100

                        MOV temp_Hi, weight_hi      ; save in temp

                        MOV temp_lo, weight_lo      ; save in temp

                        MOV weight_hi, remainder_hi ;load dremainder from first division

                        MOV weight_lo, remainder_lo ;

                        MOV count,#99               ; go *100

                        ACALL mul_Weight_100_10     ;

                        ACALL 16_bit_div

                        MOV A, R2                          ; add (Qh:Ql)*100 to (Rh:Rl)*100/height
```

```
ADD A, temp_lo
MOV temp_lo,A
CLR A
ADDC A, R3
ADDC A, temp_Hi
MOV temp_hi, A
MOV weight_hi, temp_hi          ; perform second division
MOV weight_lo, temp_lo          ;
ACALL 16_bit_div                ;perform seccond division by height
MOV remainder_hi, R1
MOV remainder_lo, R0
MOV weight_hi, R3
MOV weight_lo, R2
MOV count,#99
ACALL mul_Weight_100_10
MOV temp_hi, weight_hi
MOV temp_lo, weight_lo
MOV weight_hi,remainder_hi
MOV weight_lo, remainder_lo
MOV count,#99
ACALL mul_weight_100_10
ACALL 16_bit_div
MOV A,R2
ADD A, temp_lo
MOV temp_lo,A
CLR A
ADDC A,R3
ADDC A, temp_hi
```

```
                MOV temp_hi,A
                MOV count,#0ffh
                MOV weight_hi,#0
                MOV weight_lo,temp_hi
                ACALL mul_weight_100_10
                MOV A, weight_lo
                ADD A, temp_lo
                MOV weight_lo, A
                CLR A
                ADDC A, weight_hi
                MOV weight_hi,A
                RET
;*******************************************************************************

16_bit_div:     MOV B,#00h
                MOV R5,#0
                MOV R4,#0
                MOV R1, weight_hi
                MOV R0, weight_lo
                MOV R3, height_hi
                MOV R2, height_lo
div1:           INC B
                MOV A, R2
                RLC A
                MOV R2, A
                MOV A, R3
                RLC A
                MOV R3, A
                JNC div1
div2:           MOV A, R3
                RRC A
                MOV R3, A
                MOV A, R2
                RRC A
                MOV R2, A
                CLR C
                MOV 07h, R1
                MOV 06h, R0
                MOV A, R0
                SUBB A, R2
                MOV R0, A
                MOV A, R1
                SUBB A, R3
                MOV R1, A
                JNC div3
                MOV R1,07h
                MOV R0, 06h
div3:           CPL C
                MOV A, R4
                RLC A
                MOV R4, A
                MOV A, R5
                RLC A
                MOV R5, A
                DJNZ B, div2
                MOV R3, 05h
```

```
                    MOV R2, 04;

                    RET


;************;*****************************************************

scan_key:       ACALL read_key_code

                    RET


;*********************************************************

;lowest level routine that handles keypad decoding

read_key_code:  MOV key_code,#10h

                    CLR key_valid

                    JB row_1, skip_1

                    MOV key_code,#0

skip_1:         JB row_2, skip_2

                    MOV key_code,#4

skip_2:         JB row_3, skip_3

                    MOV key_code,#8

skip_3:         JB row_4, flip_bits

                    MOV key_code,#12

flip_bits:      MOV p3,#11110000b

                    ACALL settle_Delay

                    JB col_1, skip_5

                    MOV temp,#0

skip_5:         JB col_2, skip_6

                    MOV temp,#1

skip_6:         JB col_3.skip_7

                    MOV temp,#2

skip_7:         JB col_4,compute_key
```

```
                        MOV temp,#3
compute_key:        MOV p3,#00001111b
                        MOV A, ke _code
                        XRL A,#10h
                        JZ no_key
                        MOV A, key_code
                        ADD A,temp
                        MOV DPTR,#xlate_table2
                        MOVC A,@a+dptr
                        SETB key_Valid
no_key:             RET


xlate_table2:       DB 1,2,3,15,4,5,6,14,7,8,9,13,10,0,11,12


settle_delay:       MOV count4,#0
                        DJNZ count4,$
                        RET


;*******************************************************************
;R7......R3
Hex2BCD:
    MOV R1,weight_hi
        MOV R2, weight_lo
    MOV R3,#00D
    MOV R4,#00D
    MOV R5,#00D
    MOV R6,#00D
    MOV R7,#00D
```

```
        MOV B,#10D
    MOV A,R2
    DIV AB
     MOV  R3,B
      MOV  B,#10        ; R7,R6,R5,R4,R3
       DIV  AB
       MOV  R4,B
       MOV  R5,A
       CJNE R1,#0H,HIGH_BYTE  ; CHECK FOR HIGH BYTE
       SJMP ENDD

HIGH_BYTE:      MOV  A,#6
          ADD  A,R3
                MOV  B,#10
                DIV  AB
          MOV  R3,B
                ADD  A,#5
          ADD  A,R4
                MOV  B,#10
                DIV  AB
          MOV  R4,B
                ADD  A,#2
          ADD  A,R5
                MOV  B,#10
                DIV  AB
          MOV  R5,B
          CJNE R6,#00D,ADD_IT
          SJMP CONTINUE
```

```
                        RET

;*****************************************************

show_height_now:    MOV dp_pos1,#01111111b

                        MOV dp_pos2,#0ffh

                        MOV data_0, height_0_temp

                        MOV data_1, height_1_temp

                        MOV data_2, height_2_temp

                        ACALL write_key

                        RET

;*****************************************************

delay_2_show:   MOV count3,#0

                        DJNZ count3,$

                        RET
;*****************************************************


xlate_Table:        DB 11000000b,11111001b,10100100b,10110000b,10011001b,10010010b,10000010b,11111000b,10000000b,10010000b

;*****************************************************************

reset_Delay:        MOV R7,#0

reload:             MOV R6,#0

                    DJNZ R6,$

                    DJNZ R7, reload

                    RET
;*****************************************************

write_now:              MOV dataport,data_0

                        CLR p2.0
```

```
        MOV B,#10D

    MOV A,R2

    DIV AB

    MOV  R3,B            ;

    MOV  B,#10           ; R7,R6,R5,R4,R3

    DIV  AB

    MOV  R4,B

    MOV  R5,A

    CJNE R1,#0H,HIGH_BYTE   ; CHECK FOR HIGH BYTE

    SJMP ENDD


HIGH_BYTE:      MOV  A,#6

        ADD  A,R3

                MOV  B,#10

                DIV  AB

        MOV  R3,B

                ADD  A,#5

        ADD  A,R4

                MOV  B,#10

                DIV  AB

        MOV  R4,B

                ADD  A,#2

        ADD  A,R5

                MOV  B,#10

                DIV  AB

        MOV  R5,B

        CJNE R6,#00D,ADD_IT

        SJMP CONTINUE
```

```
ADD_IT:
        ADD A,R6
CONTINUE:
        MOV R6,A
        DJNZ R1,HIGH_BYTE
        MOV B, #10D
        MOV A,R6
        DIV AB
        MOV R6,B
        MOV R7,A
        ACALL chk_pos
 endD:  RET
;************************************************************************

 write_display:    JB weight_ptr, show_Weight_now
                   JB height_ptr, show_height_now

 show_result:      MOV data_0, result_0
                   MOV data_1, result_1
                   MOV data_2, result_2
                   ACALL write_key
                   RET
  ;*************************************************************

  show_Weight_now:  MOV data_0, weight_0_temp
                    MOV dp_pos1,#0ffh
                    MOV dp_pos2,#0ffh
                    MOV data_1, weight_1_temp
                    MOV data_2, weight_2_temp
                    ACALL write_key
```

```
                ACALL delay_2_show
                SETB p2.0
                MOV dataport, data_1
                CLR p2.1
                ACALL delay_2_show
                SETB p2.1
                MOV dataport, data_2
                CLR p2.2
                ACALL delay_2_show
                SETB p2.2
                RET
;*****************************************************************
write_key:      MOV DPTR,#xlate_table
                MOV A, data_0
                XRL A, #0ffh
                JZ skip_a
                XRL A,#0ffh
                MOVC A,@a+dptr
                ANL A, dp_pos1
                MOV data_0,A
skip_a:         MOV A, data_1
                XRL A, #0ffh
                JZ skip_c
                XRL A,#0ffh
                MOVC A,@a+dptr
                ANL A, dp_pos2
                MOV data_1, A
skip_c:
```

```
;***********************************************************************************8

;note:                    max weight:255

;mn height: 0.63
```