

Embedded System & Microcontroller Application (4351102) - Summer 2024 Solution Gujarati

Milav Dabgar

May 16, 2024

પ્રશ્ન 1(અ) [3 ગુણ]

એમ્બેડેડ સિસ્ટમની વ્યાખ્યા શું છે? એમ્બેડેડ સિસ્ટમનું ઉદાહરણ આપો.

જવાબ

એમ્બેડેડ સિસ્ટમ એ એક વિશેષ કમ્પ્યુટર સિસ્ટમ છે જે ચોક્કસ કાર્યો કરવા માટે સમર્પિત કાર્યો સાથે ડિઝાઇન કરવામાં આવે છે. તે હાર્ડવેર અને સોફ્ટવેર ઘટકોને જોડે છે જે વિશાળ સિસ્ટમમાં એકીકૃત થાય છે.

મુખ્ય લક્ષણો:

- રીઅલ-ટાઇમ ઓપરેશન: નિર્દિષ્ટ સમય મર્યાદામાં ઇનપુટ્સનો પ્રતિસાદ આપે છે
- સમર્પિત કાર્ય: ચોક્કસ એપ્લિકેશન માટે ડિઝાઇન કરેલું
- રિસોર્સ મર્યાદાઓ: મર્યાદિત મેમરી, પાવર અને પ્રોસેસિંગ ક્ષમતાઓ

ઉદાહરણ: વોશિંગ મશીન કન્ટ્રોલર જે વોશ સાઇકલ્સ, પાણીનું તાપમાન અને ટાઇમિંગને આપમેળે મેનેજ કરે છે.

મેમરી ટ્રીક

“SMART Embedded: Specialized, Microprocessor-based, Application-specific, Real-time, Task-oriented”

પ્રશ્ન 1(બ) [4 ગુણ]

રીઅલ-ટાઇમ ઓપરેટિંગ સિસ્ટમ (RTOS) ને વ્યાખ્યાયિત કરો અને RTOS ની ત્રણ લાક્ષણિકતાઓની યાદી બનાવો.

જવાબ

RTOS એ એક ઓપરેટિંગ સિસ્ટમ છે જે રીઅલ-ટાઇમ એપ્લિકેશન્સને હેન્ડલ કરવા માટે ડિઝાઇન કરવામાં આવ્યું છે જ્યાં સિસ્ટમ ઓપરેશન માટે ટાઇમિંગ અવરોધો નિર્ણાયક છે.

કોષ્ટક 1. RTOS લાક્ષણિકતાઓ

લાક્ષણિકતા	વર્ણન
નિર્ધારિત પ્રતિસાદ	નિર્ણાયક કાર્યો માટે ગેરંટીડ રિસ્પોન્સ ટાઇમ
પ્રાથમિકતા-આધારિત શેડ્યુલિંગ	ઉચ્ચ પ્રાથમિકતાના કાર્યો નીચા પ્રાથમિકતાના કાર્યો પહેલાં ચાલે છે
મલ્ટિટાસ્કિંગ સપોર્ટ	બહુવિધ કાર્યો એકસાથે ચાલી શકે છે

વધારાની વિશેષતાઓ:

- કાર્ય વ્યવસ્થાપન: બહુવિધ સમાંતર પ્રક્રિયાઓને અસરકારક રીતે હેન્ડલ કરે છે
- ઇન્ટરપ્ટ હેન્ડલિંગ: બાહ્ય ઘટનાઓને ઝડપી પ્રતિસાદ
- મેમરી વ્યવસ્થાપન: એમ્બેડેડ એપ્લિકેશન્સ માટે ઓપ્ટિમાઇઝ્ડ

મેમરી ટ્રીક

“DPM RTOS: Deterministic, Priority-based, Multitasking”

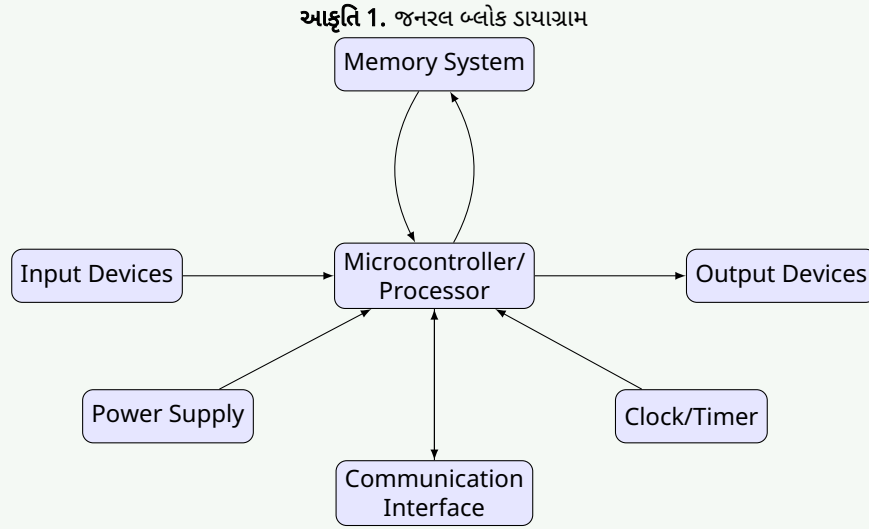
પ્રશ્ન 1(ક) [7 ગુણ]

અ) એમ્બેડેડ સિસ્ટમનો જનરલ બ્લોક ડાયાગ્રામ દોરો

બ) એમ્બેડેડ સિસ્ટમ માટે માઇક્રોકન્ટ્રોલર પસંદ કરવાના માપદંડો સમજાવો.

જવાબ

અ) જનરલ બ્લોક ડાયાગ્રામ:



બ) માઇક્રોકન્ટ્રોલર પસંદગીના માપદંડો:

કોષ્ટક 2. પસંદગીના માપદંડો

માપદંડ	વિચારણાઓ
પ્રોસેસિંગ સ્પીડ	ક્લોક ફ્રીક્વન્સી, ઇન્સ્ટ્રક્શન એક્ઝિક્યુશન ટાઇમ
મેમરી જરૂરિયાતો	Flash, RAM, EEPROM ક્ષમતા
I/O ક્ષમતાઓ	પિન્સની સંખ્યા, વિશેષ કાર્યો
પાવર વપરાશ	બેટરી લાઇફ, સ્લીપ મોડ્સ
કિંમત	બજેટ અવરોધો, વોલ્યુમ પ્રાઇસિંગ
ડેવલપમેન્ટ ટૂલ્સ	કમ્પાઇલર, ડિબગર ઉપલબ્ધતા

મુખ્ય પરિબલો:

- પ્રદર્શન આવશ્યકતાઓ: પ્રોસેસિંગ સ્પીડ અને રીઅલ-ટાઇમ અવરોધો
- ઇન્ટરફેસ જરૂરિયાતો: ADC, PWM, કમ્યુનિકેશન પ્રોટોકોલ્સ
- પર્યાવરણીય પરિસ્થિતિઓ: ઓપરેટિંગ તાપમાન, ભેજ

મેમરી ટ્રીક

“PMPICD Selection: Performance, Memory, Power, Interface, Cost, Development tools”

OR

પ્રશ્ન 1(ક) [7 ગુણ]

ATmega32 ની પિન ગોઠવણી સમજાવો.

જવાબ

ATmega32 એ 40-પિન માઇક્રોકન્ટ્રોલર છે જેમાં ચાર 8-બિટ I/O પોર્ટ્સ અને વિવિધ વિશેષ કાર્યાત્મક પિન્સ છે. પોર્ટ ગોઠવણી:

કોષ્ટક 3. પોર્ટ ગોઠવણી

પોર્ટ	પિન્સ	કાર્યો
Port A	PA0-PA7	ADC ચેનલ્સ, જનરલ I/O
Port B	PB0-PB7	SPI, PWM, બાહ્ય ઇન્ટરપ્ટ્સ
Port C	PC0-PC7	TWI, જનરલ I/O
Port D	PD0-PD7	USART, બાહ્ય ઇન્ટરપ્ટ્સ, PWM

વિશેષ પિન્સ:

- VCC/GND: પાવર સપ્લાઇ પિન્સ
- AVCC/AGND: ADC માટે એનાલોગ પાવર સપ્લાઇ
- XTAL1/XTAL2: ક્રિસ્ટલ ઓસિલેટર કનેક્શન્સ
- RESET: એક્ટિવ લો રીસેટ ઇનપુટ
- AREF: ADC રેફરન્સ વોલ્ટેજ

પિન કાર્યો:

- ડ્યુઅલ-પર્પઝ પિન્સ: મોટાભાગની પિન્સમાં વૈકલ્પિક કાર્યો છે
- ઇનપુટ/આઉટપુટ ક્ષમતા: બધી પોર્ટ પિન્સ ટ્રિફેશીય છે
- આંતરિક પુલ-અપ: ઇનપુટ પિન્સ માટે સોફ્ટવેર રૂપરેખાંકિત

મેમરી ટ્રીક

"ABCD Ports: ADC, Bus interfaces, Communication, Data transfer"

પ્રશ્ન 2(અ) [3 ગુણ]

ATMEGA32 નું ડેટા મેમરી આર્કિટેક્ચર સમજાવો.

જવાબ

ATmega32 ડેટા મેમરી ત્રણ વિભાગોનો સમાવેશ કરે છે જે એકીકૃત સરનામા સ્થળમાં આયોજિત છે. મેમરી સંગઠન:

કોષ્ટક 4. મેમરી સંગઠન

ત્રણ વિભાગો	સરનામા શ્રેણી	કદ	હેતુ
જનરલ રજિસ્ટર્સ	0x00-0x1F	32 બાઇટ્સ	વર્કિંગ રજિસ્ટર્સ R0-R31
I/O રજિસ્ટર્સ	0x20-0x5F	64 બાઇટ્સ	કન્ટ્રોલ અને સ્ટેટસ રજિસ્ટર્સ
આંતરિક SRAM	0x60-0x45F	2048 બાઇટ્સ	ડેટા સ્ટોરેજ અને સ્ટેક

મુખ્ય વિશેષતાઓ:

- એકીકૃત એડ્રેસિંગ: બધી મેમરી એક સરનામા સ્થળ દ્વારા સુલભ
- રજિસ્ટર ફાઇલ: અંકગણિત અને તર્ક ઓપરેશન્સ માટે R0-R31
- સ્ટેક પોઇન્ટર: SRAM માં સ્ટેકની ટોપ તરફ નિર્દેશ કરે છે

મેમરી ટ્રીક

“GIS Memory: General registers, IO registers, SRAM”

પ્રશ્ન 2(બ) [4 ગુણ]

પ્રોગ્રામ સ્ટેટસ વર્ડ સમજાવો.

જવાબ

SREG (સ્ટેટસ રજિસ્ટર) માં ફ્લેગ્સ છે જે અંકગણિત અને તર્ક ઓપરેશન્સના પરિણામને પ્રતિબિંબિત કરે છે.
SREG બિટ રૂપરેખાંકન:

કોષ્ટક 5. SREG બિટ રૂપરેખાંકન

બિટ	ફ્લેગ	વર્ણન
બિટ 7	I	ગ્લોબલ ઇન્ટરપ્ટ એનેબલ
બિટ 6	T	બિટ કોપી સ્ટોરેજ
બિટ 5	H	હાફ કેરી ફ્લેગ
બિટ 4	S	સાઇન ફ્લેગ
બિટ 3	V	ઓવરફ્લો ફ્લેગ
બિટ 2	N	નેગેટિવ ફ્લેગ
બિટ 1	Z	ઝીરો ફ્લેગ
બિટ 0	C	કેરી ફ્લેગ

ફ્લેગ કાર્યો:

- અંકગણિત ઓપરેશન્સ: C, Z, N, V, H ફ્લેગ્સ આપમેળે અપડેટ થાય છે
- શરતી બ્રાન્ચિંગ: નિર્ણય લેવા માટે ફ્લેગ્સનો ઉપયોગ
- ઇન્ટરપ્ટ નિયંત્રણ: I ફ્લેગ ગ્લોબલ ઇન્ટરપ્ટ્સને સક્ષમ/અક્ષમ કરે છે

મેમરી ટ્રીક

“I THSVNZC: Interrupt, Transfer, Half-carry, Sign, oVerflow, Negative, Zero, Carry”

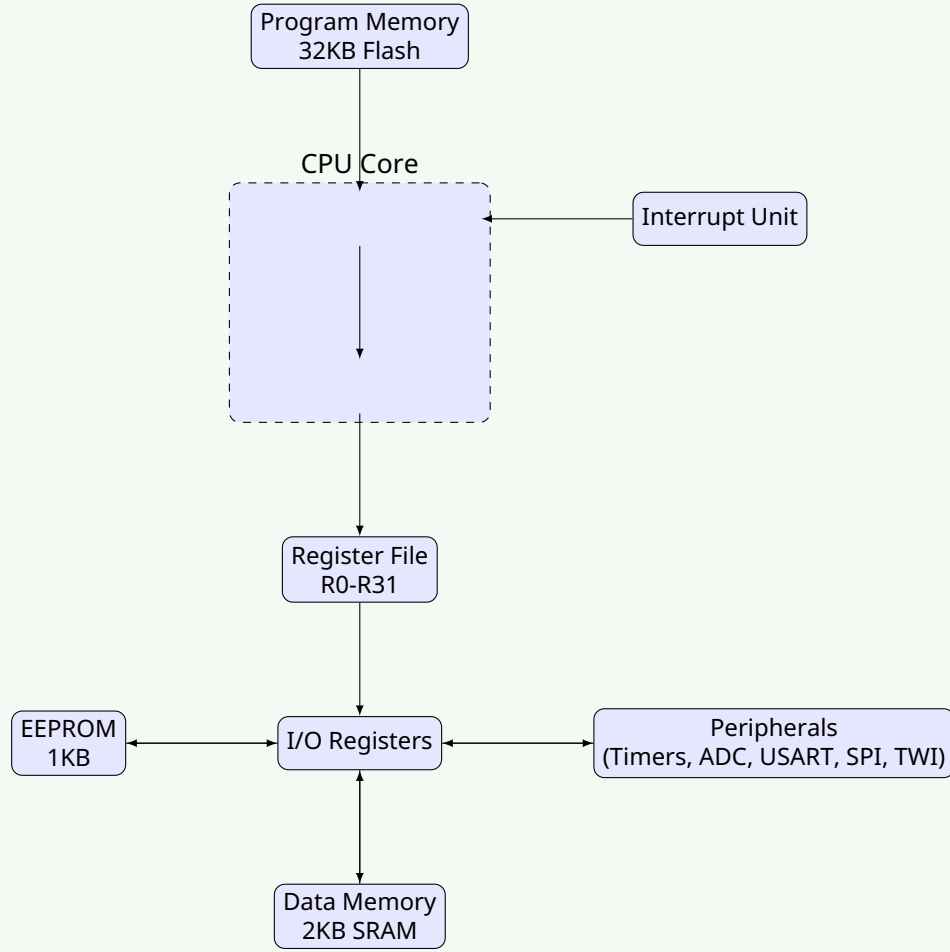
પ્રશ્ન 2(ક) [7 ગુણ]

ATMEGA32 ના આર્કિટેક્ચર દોરો અને સમજાવો.

જવાબ

ATmega32 આર્કિટેક્ચર:

આકૃતિ 2. ATmega32 આર્કિટેક્ચર



આર્કિટેક્ચર ઘટકો:

કોષ્ટક 6. આર્કિટેક્ચર ઘટકો

ઘટક	વર્ણન
હાર્વર્ડ આર્કિટેક્ચર	અલગ પ્રોગ્રામ અને ડેટા મેમરી બસ
RISC કોર	131 સૂચનાઓ, મોટાભાગે સિંગલ-સાઇકલ એક્ઝિક્યુશન
ALU	8-બિટ અંકગણિત અને તર્ક ઓપરેશન્સ
રજિસ્ટર ફાઇલ	32 × 8-બિટ વર્કિંગ રજિસ્ટર્સ

મેમરી સિસ્ટમ:

- પ્રોગ્રામ મેમરી: સૂચનાઓ સંગ્રહ કરવા માટે 32KB Flash
- ડેટા મેમરી: ચલો અને સ્ટેક માટે 2KB SRAM
- EEPROM: 1KB નોન-વોલેટાઇલ ડેટા સ્ટોરેજ

પેરિફરલ વિશેષતાઓ:

- ત્રણ ટાઇમર/કાઉન્ટર્સ: 8-બિટ અને 16-બિટ ટાઇમર્સ
- 8-ચેનલ ADC: 10-બિટ રીઝોલ્યુશન
- કમ્યુનિકેશન ઇન્ટરફેસ: USART, SPI, TWI

મેમરી ટ્રીક

“HRAM Micro: Harvard architecture, RISC core, ALU, Memory system”

OR

પ્રશ્ન 2(અ) [3 ગુણ]

ATMEGA32 ના પ્રોગ્રામ કાઉન્ટર સમજાવો.

જવાબ

પ્રોગ્રામ કાઉન્ટર (PC) એ 16-બિટ રજિસ્ટર છે જે એક્ઝિક્યુટ થવાની આગલી સૂચનાનું સરનામું ધરાવે છે.
PC લાક્ષણિકતાઓ:

કોષ્ટક 7. PC લાક્ષણિકતાઓ

વિશેષતા	વર્ણન
કદ	16-બિટ (64KB પ્રોગ્રામ મેમરીને સરનામું આપી શકે છે)
રીસેટ વેલ્યુ	0x0000 (શરૂઆતથી એક્ઝિક્યુશન શરૂ કરે છે)
વૃદ્ધિ	સૂચના મેળવ્યા પછી આપમેળે વધે છે
જમ્પ/બ્રાન્ચ	જમ્પ, બ્રાન્ચ અને કોલ સૂચનાઓ દ્વારા સુધારેલું

PC ઓપરેશન્સ:

- અનુક્રમિક એક્ઝિક્યુશન: મોટાભાગની સૂચનાઓ માટે PC 1 વધે છે
- બ્રાન્ચ સૂચનાઓ: PC ને ટાર્ગેટ એડ્રેસ સાથે લોડ કરવામાં આવે છે
- ઇન્ટરપ્ટ હેન્ડલિંગ: PC સ્ટેકમાં સાચવાય છે, ઇન્ટરપ્ટ વેક્ટર સાથે લોડ કરાય છે

મેમરી ટ્રીક

“SRIB PC: Sequential, Reset, Increment, Branch”

OR

પ્રશ્ન 2(બ) [4 ગુણ]

AVR માઇક્રોકન્ટ્રોલરમાં કલોક અને રીસેટ સર્કિટની ભૂમિકા સમજાવો.

જવાબ

કલોક સિસ્ટમ:

કોષ્ટક 8. કલોક સ્ત્રોત

કલોક સ્ત્રોત	વર્ણન
બાહ્ય ક્રિસ્ટલ	ઉચ્ચ ચોકસાઈ, 1-16 MHz સામાન્ય
આંતરિક RC	બિલ્ટ-ઇન 8 MHz ઓસિલેટર
બાહ્ય કલોક	બાહ્ય કલોક સિગ્નલ ઇનપુટ
લો-ફ્રીક્વન્સી ક્રિસ્ટલ	RTC એપ્લિકેશન્સ માટે 32.768 kHz

રીસેટ સર્કિટ કાર્યો:

- પાવર-ઓન રીસેટ: પાવર લાગુ થયા પછી આપમેળે રીસેટ
- બ્રાઉન-આઉટ રીસેટ: સપ્લાઇ વોલ્ટેજ ઘટે છે ત્યારે રીસેટ
- બાહ્ય રીસેટ: RESET પિન દ્વારા મેન્યુઅલ રીસેટ
- વોચડોગ રીસેટ: વોચડોગ ટાઇમર ટાઇમઆઉટથી રીસેટ

મુખ્ય વિશેષતાઓ:

- કલોક વિતરણ: સિસ્ટમ કલોક CPU અને પેરિફેરલ્સ યલાવે છે
- રીસેટ ક્રમ: બધા રજિસ્ટર્સને ડિફોલ્ટ વેલ્યુમાં પ્રારંભ કરે છે
- ફ્યુઝ બિટ્સ: કલોક સ્ત્રોત અને રીસેટ વિકલ્પો રૂપરેખાંકિત કરે છે

મેમરી ટ્રીક

“CEIL Clock: Crystal, External, Internal, Low-frequency”

OR

પ્રશ્ન 2(ક) [7 ગુણ]

TCCRn અને TIFR ટાઇમર રજિસ્ટર સમજાવો

જવાબ

TCCRn (ટાઇમર/કાઉન્ટર કન્ટ્રોલ રજિસ્ટર):

કોષ્ટક 9. TCCRn રજિસ્ટર્સ

રજિસ્ટર	કાર્ય
TCCR0	Timer0 ઓપરેશન મોડ નિયંત્રિત કરે છે
TCCR1A/B	Timer1 (16-બિટ) ઓપરેશન નિયંત્રિત કરે છે
TCCR2	Timer2 ઓપરેશન મોડ નિયંત્રિત કરે છે

TCCR બિટ કાર્યો:

- કલોક સિલેક્ટ (CS): કલોક સ્રોત અને પ્રીસ્કેલર પસંદ કરે છે
- વેવફોર્મ જનરેશન (WGM): ટાઇમર મોડ સેટ કરે છે (Normal, CTC, PWM)
- કમ્પેર આઉટપુટ મોડ (COM): આઉટપુટ પિન વર્તન નિયંત્રિત કરે છે

TIFR (ટાઇમર ઇન્ટરપ્ટ ફ્લેગ રજિસ્ટર):

કોષ્ટક 10. TIFR ફ્લેગ્સ

બિટ	ફ્લેગ	વર્ણન
TOV	ટાઇમર ઓવરફ્લો	ટાઇમર ઓવરફ્લો થાય છે ત્યારે સેટ થાય છે
OCF	આઉટપુટ કમ્પેર	કમ્પેર મેચ થાય છે ત્યારે સેટ થાય છે
ICF	ઇનપુટ કેપ્ચર	ઇનપુટ કેપ્ચર ઇવેન્ટ થાય છે ત્યારે સેટ થાય છે

ટાઇમર ઓપરેશન્સ:

- મોડ પસંદગી: Normal, CTC, Fast PWM, Phase Correct PWM
- ઇન્ટરપ્ટ જનરેશન: સક્ષમ હોય ત્યારે ફ્લેગ્સ ઇન્ટરપ્ટ ટ્રિગર કરે છે
- આઉટપુટ જનરેશન: મોટર કન્ટ્રોલ, LED ડિમિંગ માટે PWM સિગ્નલ્સ

મેમરી ટ્રીક

“TCCR WGM: Timer Control, Clock, Register, Waveform Generation Mode”

પ્રશ્ન 3(અ) [3 ગુણ]

C માં પ્રોગ્રામિંગ AVR માટે વિવિધ ડેટા ટાઇપ અલગ પાડો

જવાબ

AVR C ડેટા ટાઇપ્સ:

કોષ્ટક 11. AVR C ડેટા ટાઇપ્સ

ડેટા ટાઇપ	કદ	શ્રેણી	ઉપયોગ
char	8-બિટ	-128 to 127	અક્ષરો, નાના પૂર્ણાંકો
unsigned char	8-બિટ	0 to 255	પોર્ટ મૂલ્યો, ફ્લેગ્સ
int	16-બિટ	-32768 to 32767	સામાન્ય પૂર્ણાંકો
unsigned int	16-બિટ	0 to 65535	કાઉન્ટર્સ, સરનામાઓ
long	32-બિટ	-2^{31} to $2^{31}-1$	મોટી ગણતરીઓ
float	32-બિટ	$\pm 3.4 \times 10^{38}$	દશાંશ ગણતરીઓ

વિશેષ વિચારણાઓ:

- મેમરી કાર્યક્ષમ: સૌથી નાની યોગ્ય ડેટા ટાઇપનો ઉપયોગ કરો
- પોર્ટ ઓપરેશન્સ: 8-બિટ પોર્ટ્સ માટે unsigned char
- ટાઇમિંગ ગણતરીઓ: ટાઇમર મૂલ્યો માટે unsigned int

મેમરી ટ્રીક

“CUIL Float: Char, Unsigned, Int, Long, Float”

પ્રશ્ન ૩(બ) [4 ગુણ]

પોર્ટ C ના તમામ બિટ્સને 200 વખત ટોગલ કરવા માટે C પ્રોગ્રામ લખો.

જવાબ

```

1 #include <avr/io.h>
2 #include <util/delay.h>
3
4 int main() {
5     DDRC = 0xFF; // Set Port C as output
6     unsigned int count = 0;
7
8     while(count < 200) {
9         PORTC = 0xFF; // Set all bits high
10        _delay_ms(100); // Delay
11        PORTC = 0x00; // Set all bits low
12        _delay_ms(100); // Delay
13        count++; // Increment counter
14    }
15    return 0;
16 }
```

પ્રોગ્રામ સમજૂતી:

- DDRC = 0xFF: પોર્ટ C ના બધા પિન્સને આઉટપુટ તરીકે રૂપરેખાંકિત કરે છે
- ટોગલ ઓપરેશન: 0xFF અને 0x00 વચ્ચે ફેરબદલી કરે છે
- કાઉન્ટર: ટોગલ સાઇકલ્સની સંખ્યા ટ્રેક કરે છે
- વિલંબ: ટોગલ ઓપરેશન માટે દૃશ્યમાન ટાઇમિંગ પ્રદાન કરે છે

મેમરી ટ્રીક

“DTC Loop: DDR setup, Toggle bits, Count iterations, Loop control”

પ્રશ્ન ૩(ક) [7 ગુણ]

અ) LED PORTB ના પિન સાથે જોડાયેલ છે. LED પર 0 થી FFh સુધીની ગણતરી બતાવવા માટે AVR પ્રોગ્રામ્સ લખો
 બ) પોર્ટ C માંથી ડેટાનો બાઇટ મેળવવા માટે AVR C પ્રોગ્રામ લખો. જો તે 100 કરતા ઓછો હોય તો તેને પોર્ટ B પર મોકલો; નહીંતર, તેને પોર્ટ D પર મોકલો.

જવાબ

અ) બાઇનરી કાઉન્ટર ડિસ્પ્લે:

```
1 #include <avr/io.h>
2 #include <util/delay.h>
3
4 int main() {
5     DDRB = 0xFF;    // Port B as output
6     unsigned char count = 0;
7
8     while(1) {
9         PORTB = count; // Display count on LEDs
10        _delay_ms(500); // Delay for visibility
11        count++;        // Increment counter
12        if(count > 0xFF) // Reset after 255
13            count = 0;
14    }
15    return 0;
16 }
```

બ) શરતી ડેટા ટ્રાન્સફર:

```
1 #include <avr/io.h>
2
3 int main() {
4     DDRC = 0x00;    // Port C as input
5     DDRB = 0xFF;    // Port B as output
6     DDRD = 0xFF;    // Port D as output
7
8     while(1) {
9         unsigned char data = PINC; // Read from Port C
10
11        if(data < 100) {
12            PORTB = data;    // Send to Port B
13            PORTD = 0x00;    // Clear Port D
14        } else {
15            PORTD = data;    // Send to Port D
16            PORTB = 0x00;    // Clear Port B
17        }
18    }
19    return 0;
20 }
```

મુખ્ય પ્રોગ્રામિંગ વિભાવનાઓ:

- પોર્ટ દિશા: DDR રજિસ્ટર્સ ઇનપુટ/આઉટપુટ રૂપરેખાંકિત કરે છે
- ડેટા વાંચવું: PIN રજિસ્ટર્સ ઇનપુટ મૂલ્યો વાંચે છે
- શરતી તર્ક: નિર્ણય લેવા માટે if-else નિવેદનો

મેમરી ટ્રીક

“RCC Data: Read input, Compare value, Conditional output”

OR

પ્રશ્ન 3(અ) [3 ગુણ]

-3 થી +3 પોર્ટ B ની કિંમતો મોકલવા માટે AVR C પ્રોગ્રામ લખો

જવાબ

```

1 #include <avr/io.h>
2 #include <util/delay.h>
3
4 int main() {
5     DDRB = 0xFF; // Port B as output
6     signed char values[] = {-3, -2, -1, 0, 1, 2, 3};
7     unsigned char i = 0;
8
9     while(1) {
10        PORTB = values[i]; // Send value to Port B
11        _delay_ms(1000); // 1 second delay
12        i++; // Next value
13        if(i > 6) i = 0; // Reset index
14    }
15    return 0;
16 }
```

પ્રોગ્રામ વિશેષતાઓ:

- સાઇન ડેટા: નકારાત્મક મૂલ્યો માટે signed char નો ઉપયોગ
- અરે સ્ટોરેજ: સરળ પહોંચ માટે અરેમાં મૂલ્યો સંગ્રહ
- ચક્રીય ઓપરેશન: તમામ મૂલ્યો દ્વારા સતત ચક્ર

મેમરી ટ્રીક

“SAC Values: Signed char, Array storage, Cyclic operation”

OR

પ્રશ્ન 3(બ) [4 ગુણ]

ASCII અક્ષરો 0,1,2,3,4,5,A,B,C અને D માટે હેક્સ મૂલ્યો પોર્ટ B પર મોકલવા માટે AVR C પ્રોગ્રામ લખો.

જવાબ

```

1 #include <avr/io.h>
2 #include <util/delay.h>
3
4 int main() {
5     DDRB = 0xFF; // Port B as output
6
7     // ASCII hex values array
8     unsigned char ascii_values[] = {
9         0x30, // '0'
10        0x31, // '1'
11        0x32, // '2'
12        0x33, // '3'
13        0x34, // '4'
14        0x35, // '5'
15        0x41, // 'A'
16        0x42, // 'B'
17        0x43, // 'C'
18    };
19 }
```

```

18     0x44 // 'D'
19 };
20
21 unsigned char i = 0;
22
23 while(1) {
24     PORTB = ascii_values[i]; // Send ASCII value
25     _delay_ms(500);         // Delay
26     i++;                     // Next character
27     if(i > 9) i = 0;         // Reset index
28 }
29 return 0;
30 }

```

ASCII મૂલ્યો ટેબલ:

કોષ્ટક 12. ASCII મૂલ્યો

અક્ષર	હેક્સ મૂલ્ય	બાઇનરી
'0'	0x30	00110000
'1'	0x31	00110001
'A'	0x41	01000001
'B'	0x42	01000010

મેમરી ટ્રીક

“HAC ASCII: Hex values, Array storage, Cyclic transmission”

OR

પ્રશ્ન 3(ક) [7 ગુણ]

ડોર સેન્સર પોર્ટ B ના બિટ 1 સાથે જોડાયેલ છે, અને LED પોર્ટ C ના બિટ 7 સાથે જોડાયેલ છે. ડોર સેન્સર પર દેખરેખ રાખવા માટે AVR C પ્રોગ્રામ લખો અને જ્યારે તે ખુલે છે (PIN HIGH છે), LED ચાલુ કરો. ફ્લો ચાર્ટ પણ દોરો.

જવાબ

C પ્રોગ્રામ:

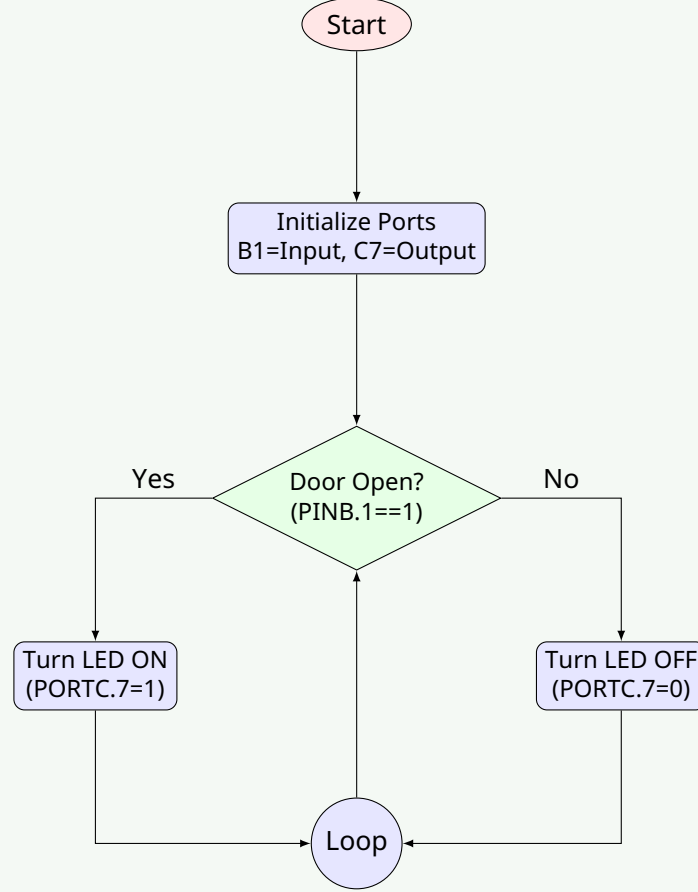
```

1 #include <avr/io.h>
2
3 int main() {
4     DDRB = 0xFD; // Port B bit 1 as input (0), others output (1)
5     DDRC = 0xFF; // Port C as output
6     PORTB = 0x02; // Enable pull-up for bit 1
7
8     while(1) {
9         if(PINB & 0x02) { // Check if door sensor is HIGH
10             PORTC |= 0x80; // Turn ON LED (bit 7)
11         } else {
12             PORTC &= 0x7F; // Turn OFF LED (bit 7)
13         }
14     }
15     return 0;
16 }

```

ફ્લો ચાર્ટ:

આકૃતિ 3. ડોર સેન્સર ફ્લો ચાર્ટ



બિટ ઓપરેશન્સ:

- ઇનપુટ વાંચવું: PINB & 0x02 બિટ 1 તપાસે છે
- LED નિયંત્રણ: PORTC |= 0x80 બિટ 7 સેટ કરે છે
- LED બંધ: PORTC &= 0x7F બિટ 7 સાફ કરે છે

મેમરી ટ્રીક

“BIC Door: Bit manipulation, Input monitoring, Conditional LED control”

પ્રશ્ન 4(અ) [3 ગુણ]

ADC માટેનું ADMUX રજિસ્ટર સમજાવો

જવાબ

ADMX (ADC મલ્ટિપ્લેક્સર સિલેક્શન રજિસ્ટર):

કોષ્ટક 13. ADMUX બિટ્સ

બિટ	નામ	વર્ણન
બિટ 7-6	REFS1:0	રેફરન્સ સિલેક્શન
બિટ 5	ADLAR	ADC લેફ્ટ એડજસ્ટ રિઝલ્ટ
બિટ 4-0	MUX4:0	એનાલોગ ચેનલ સિલેક્શન

રેફરન્સ સિલેક્શન (REFS1:0):

- 00: AREF, આંતરિક Vref બંધ
- 01: AREF પિન પર બાહ્ય કેપેસિટર સાથે AVCC
- 10: આરક્ષિત
- 11: આંતરિક 2.56V રેફરન્સ

ચેનલ સિલેક્શન (MUX4:0):

- 00000-00111: ADC0-ADC7 (સિંગલ-એન્ડેડ ઇનપુટ્સ)
- અન્ય કમ્બિનેશન્સ: ગ્રાઉન્ડ સાથે ડિફરન્શિયલ ઇનપુટ્સ

મેમરી ટ્રીક

“RAM ADMUX: Reference, Alignment, Multiplexer”

પ્રશ્ન 4(બ) [4 ગુણ]

વિવિધ એલસીડી પિન સમજાવો.

જવાબ

16x2 LCD પિન રૂપરેખાંકન:

કોષ્ટક 14. LCD પિન

પિન	સિમ્બોલ	કાર્ય
1	VSS	ગ્રાઉન્ડ (0V)
2	VDD	પાવર સપ્લાઇ (+5V)
3	V0	કોન્ટ્રાસ્ટ એડજસ્ટમેન્ટ
4	RS	રજિસ્ટર સિલેક્ટ (ડેટા/કમાન્ડ)
5	R/W	રીડ/રાઇટ સિલેક્ટ
6	E	એનેબલ સિગ્નલ
7-14	D0-D7	ડેટા બસ (8-બિટ)
15	A	બેકલાઇટ એનોડ (+)
16	K	બેકલાઇટ કેથોડ (-)

કન્ટ્રોલ પિન કાર્યો:

- RS = 0: કમાન્ડ રજિસ્ટર પસંદ
- RS = 1: ડેટા રજિસ્ટર પસંદ
- R/W = 0: રાઇટ ઓપરેશન
- R/W = 1: રીડ ઓપરેશન
- E: એનેબલ પલ્સ ઓપરેશન ટ્રિગર કરે છે

મેમરી ટ્રીક

“VCR EDB LCD: Vpower, Contrast, Register select, Enable, Data Bus”

પ્રશ્ન 4(ક) [7 ગુણ]

20 μ s વિલંબ સાથે PORTB ના તમામ બિટ્સને સતત ટોગલ કરવા માટે પ્રોગ્રામ લખો. વિલંબ જનરેટ કરવા માટે Timer0, નોર્મલ મોડ અને કોઈ Prescaler નો ઉપયોગ કરો

જવાબ

```

1 #include <avr/io.h>
2
3 void delay_20us() {
4     TCNT0 = 0;    // Clear timer counter
5     TCCR0 = 0x01; // No prescaler, normal mode
6     while(TCNT0 < 160); // Wait for 20us (8MHz/1 * 20us = 160)
7     TCCR0 = 0;    // Stop timer
8 }
9
10 int main() {
11     DDRB = 0xFF; // Port B as output
12
13     while(1) {
14         PORTB = 0xFF; // Set all bits high
15         delay_20us(); // 20us delay
16         PORTB = 0x00; // Set all bits low
17         delay_20us(); // 20us delay
18     }
19     return 0;
20 }

```

ટાઇમર ગણતરી:

- કલોક ફ્રીક્વન્સી: 8 MHz (ધારણા)
- ટાઇમર રીઝોલ્યુશન: $1/8\text{MHz} = 0.125\mu\text{s}$ પ્રતિ કાઉન્ટ
- જરૂરી કાઉન્ટ્સ: $20\mu\text{s} / 0.125\mu\text{s} = 160$ કાઉન્ટ્સ

Timer0 રૂપરેખાંકન:

કોષ્ટક 15. Timer0 સેટિંગ્સ

સેટિંગ	મૂલ્ય	વર્ણન
મોડ	નોર્મલ	0 થી 255 સુધી ગણો છે
Prescaler	1	કોઈ પ્રીસ્કેલિંગ નહીં
કલોક સ્ત્રોત	સિસ્ટમ કલોક	8 MHz

મેમરી ટ્રીક

``TNP Timer: Timer0, Normal mode, Prescaler none"

OR

પ્રશ્ન 4(અ) [3 ગુણ]

ટૂંકી નોંધ બે વાયર ઇન્ટરફેસ (TWI)

જવાબ

TWI (બે વાયર ઇન્ટરફેસ) - I2C પ્રોટોકોલ:
મુખ્ય વિશેષતાઓ:

કોષ્ટક 16. TWI વિશેષતાઓ

વિશેષતા	વર્ણન
બે વાયર	SDA (ડેટા) અને SCL (ક્લોક)
મલ્ટી-માસ્ટર	બહુવિધ માસ્ટર્સ બસ નિયંત્રિત કરી શકે છે
મલ્ટી-સ્લેવ	127 સુધી સ્લેવ ડિવાઇસ
સરનામું-આધારિત	7-બિટ અથવા 10-બિટ ડિવાઇસ સરનામું
દ્વિદિશીય	બંને દિશામાં ડેટા વહે છે

બસ લાક્ષણિકતાઓ:

- ઓપન-ડ્રેઇન: પુલ-અપ રજિસ્ટર્સ આવશ્યક (4.7kΩ મુળ)
- સિન્ક્રોનસ: માસ્ટર દ્વારા ક્લોક પ્રદાન કરાય છે
- સ્ટાર્ટ/સ્ટોપ કન્ડિશન્સ: કમ્યુનિકેશન માટે વિશેષ ક્રમ

મેમરી ટ્રીક

"SDA SCL TWI: Serial Data, Serial CLock, Two Wire Interface"

OR

પ્રશ્ન 4(બ) [4 ગુણ]

ADC માટેનું ADCSRA રજિસ્ટર સમજાવો

જવાબ

ADCSRA (ADC કન્ટ્રોલ અને સ્ટેટસ રજિસ્ટર A):

કોષ્ટક 17. ADCSRA રજિસ્ટર

બિટ	નામ	કાર્ય
બિટ 7	ADEN	ADC એનેબલ
બિટ 6	ADSC	ADC સ્ટાર્ટ કન્વર્ઝન
બિટ 5	ADATE	ADC ઓટો ટ્રિગર એનેબલ
બિટ 4	ADIF	ADC ઇન્ટરપ્ટ ફ્લેગ
બિટ 3	ADIE	ADC ઇન્ટરપ્ટ એનેબલ
બિટ 2-0	ADPS2:0	ADC પ્રીસ્કેલર સિલેક્ટ

પ્રીસ્કેલર સેટિંગ્સ (ADPS2:0):

કોષ્ટક 18. પ્રીસ્કેલર સેટિંગ્સ

બાઇનરી	વિભાજન પરિબલ	ADC ક્લોક (8MHz)
000	2	4 MHz
001	2	4 MHz
010	4	2 MHz
011	8	1 MHz
100	16	500 kHz
101	32	250 kHz
110	64	125 kHz
111	128	62.5 kHz

કન્ટ્રોલ કાર્યો:

- ADEN: ADC ઓપરેશન સક્ષમ કરવા માટે સેટ કરવું આવશ્યક

- **ADSC:** કન્વર્ઝન શરૂ કરવા માટે સેટ કરો, પૂર્ણ થાય ત્યારે સાફ થાય છે
- **Prescaler:** શ્રેષ્ઠ ચોકસાઈ માટે ADC ક્લોક 50-200 kHz હોવી જોઈએ

મેમરી ટ્રીક

“EASCID ADC: Enable, Auto-trigger, Start, Conversion, Interrupt, Divider”

OR

પ્રશ્ન 4(ક) [7 ગુણ]

PORTC.3 પિન પર 16 Khz ફ્રીક્વન્સીની સ્ક્વેર વેવ જનરેટ કરવા માટે પ્રોગ્રામ લખો. ક્રિસ્ટલ ફ્રીક્વન્સી 8 Mhz ધારો

જવાબ

```

1 #include <avr/io.h>
2 #include <avr/interrupt.h>
3
4 int main() {
5     // Configure PC3 as output
6     DDRC |= (1 << PC3);
7
8     // Timer1 CTC mode configuration
9     TCCR1A = 0x00;           // Normal port operation
10    TCCR1B = (1 << WGM12) | (1 << CS10); // CTC mode, no prescaler
11
12    // Calculate OCR1A value for 16 kHz
13    // Period = 1/16000 = 62.5us
14    // Half period = 31.25us
15    // OCR1A = (8MHz * 31.25us) - 1 = 249
16    OCR1A = 249;
17
18    // Enable Timer1 Compare A interrupt
19    TIMSK |= (1 << OCIE1A);
20
21    // Enable global interrupts
22    sei();
23
24    while(1) {
25        // Main loop - square wave generated by interrupt
26    }
27    return 0;
28 }
29
30 // Timer1 Compare A interrupt service routine
31 ISR(TIMER1_COMPA_vect) {
32     PORTC ^= (1 << PC3); // Toggle PC3
33 }
```

ફ્રીક્વન્સી ગણતરી:

કોષ્ટક 19. ફ્રીક્વન્સી ગણતરી

પેરામીટર	મૂલ્ય	ફોર્મ્યુલા
Target frequency	16 kHz	Given
Period	62.5 μ s	1/16000
Half period	31.25 μ s	Period/2
Timer counts	250	8MHz \times 31.25 μ s
OCR1A value	249	Counts - 1

ટાઇમર રૂપરેખાંકન:

- **Mode:** CTC (Clear Timer on Compare)
- **Prescaler:** 1 (no prescaling)
- **Interrupt:** Compare match toggles output pin

મેમરી ટ્રીક

``CTC Square: CTC mode, Timer interrupt, Compare match``

પ્રશ્ન 5(અ) [3 ગુણ]

Polling અને Interrupt વચ્ચેની તફાવત

જવાબ

Polling vs Interrupt:

કોષ્ટક 20. તફાવત

પાસું	Polling	Interrupt
CPU ઉપયોગ	સતત સ્ટેટસ તપાસે છે	ઇવેન્ટ થાય ત્યાં સુધી CPU મુક્ત
પ્રતિસાદ સમય	વેરિએબલ, પોલિંગ ફ્રીક્વન્સી પર આધારિત	ઝડપી, તાત્કાલિક પ્રતિસાદ
પાવર વપરાશ	સતત તપાસને કારણે વધારે	ઓછો, CPU સ્લીપ થઈ શકે છે
પ્રોગ્રામિંગ	સરળ, અનુક્રમિક કોડ	જટિલ, ISR ની જરૂર છે
રીઅલ-ટાઇમ	નિર્ણાયક ટાઇમિંગ માટે યોગ્ય નથી	રીઅલ-ટાઇમ સિસ્ટમો માટે ઉત્તમ

મેમરી ટ્રીક

``PIE Method: Polling inefficient, Interrupt efficient, Event-driven``

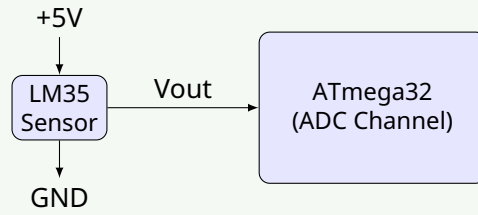
પ્રશ્ન 5(બ) [4 ગુણ]

LM35 ઇન્ટરફેસ સમજાવો.

જવાબ

LM35 તાપમાન સેન્સર ઇન્ટરફેસ:

આકૃતિ 4. LM35 ઇન્ટરફેસ



LM35 લાક્ષણિકતાઓ:

કોષ્ટક 21. LM35 લાક્ષણિકતાઓ

પેરામીટર	મૂલ્ય	વર્ણન
આઉટપુટ	10mV/°C	લીનિયર તાપમાન ગુણાંક
શ્રેણી	0°C to 100°C	ઓપરેટિંગ તાપમાન શ્રેણી
સપ્લાઇ	4V to 30V	પાવર સપ્લાઇ શ્રેણી
ચોકસાઇ	±0.5°C	તાપમાન ચોકસાઇ

ઇન્ટરફેસ કોડ:

```

1 #include <avr/io.h>
2
3 void ADC_init() {
4     ADMUX = 0x40; // AVCC reference, ADC0 channel
5     ADCSRA = 0x87; // Enable ADC, prescaler 128
6 }
7
8 unsigned int read_temperature() {
9     ADCSRA |= (1 << ADSC); // Start conversion
10    while(ADCSRA & (1 << ADSC)); // Wait for completion
11
12    // Convert ADC value to temperature
13    // Temperature = (ADC * 5000) / (1024 * 10)
14    unsigned int temp = (ADC * 5000) / 10240;
15    return temp;
16 }
  
```

મેમરી ટ્રીક

“LAC Temperature: LM35 sensor, ADC conversion, Calculation formula”

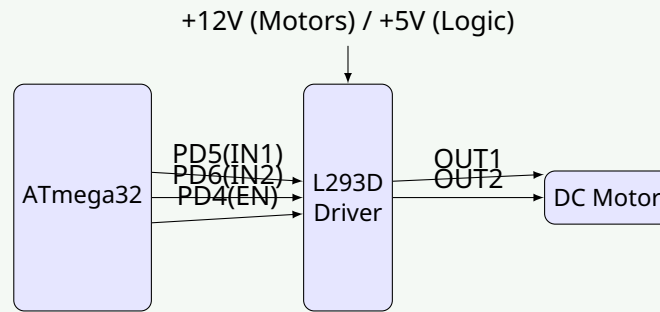
પ્રશ્ન 5(ક) [7 ગુણ]

DC મોટરને ઇન્ટરફેસ કરવા માટે પ્રોગ્રામ લખો.

જવાબ

DC મોટર ઇન્ટરફેસ સર્કિટ:

આકૃતિ 5. DC મોટર ઇન્ટરફેસ



મોટર કન્ટ્રોલ પ્રોગ્રામ:

```

1  #include <avr/io.h>
2  #include <util/delay.h>
3
4  void motor_init() {
5      DDRD |= (1 << PD4) | (1 << PD5) | (1 << PD6); // Set as output
6  }
7
8  void motor_forward() {
9      PORTD |= (1 << PD4); // Enable motor
10     PORTD |= (1 << PD5); // IN1 = 1
11     PORTD &= ~(1 << PD6); // IN2 = 0
12 }
13
14 void motor_reverse() {
15     PORTD |= (1 << PD4); // Enable motor
16     PORTD &= ~(1 << PD5); // IN1 = 0
17     PORTD |= (1 << PD6); // IN2 = 1
18 }
19
20 void motor_stop() {
21     PORTD &= ~(1 << PD4); // Disable motor
22 }
23
24 int main() {
25     motor_init();
26
27     while(1) {
28         motor_forward(); // Forward for 2 seconds
29         _delay_ms(2000);
30
31         motor_stop(); // Stop for 1 second
32         _delay_ms(1000);
33
34         motor_reverse(); // Reverse for 2 seconds
35         _delay_ms(2000);
36
37         motor_stop(); // Stop for 1 second
38         _delay_ms(1000);
39     }
40     return 0;
41 }

```

L293D ટુથ ટેબલ:

કોષ્ટક 22. L293D ટુથ ટેબલ

EN	IN1	IN2	Motor Action
0	X	X	Stop
1	0	0	Stop
1	0	1	Reverse
1	1	0	Forward
1	1	1	Stop

મેમરી ટ્રીક

“LED Motor: L293D driver, Enable control, Direction pins”

OR

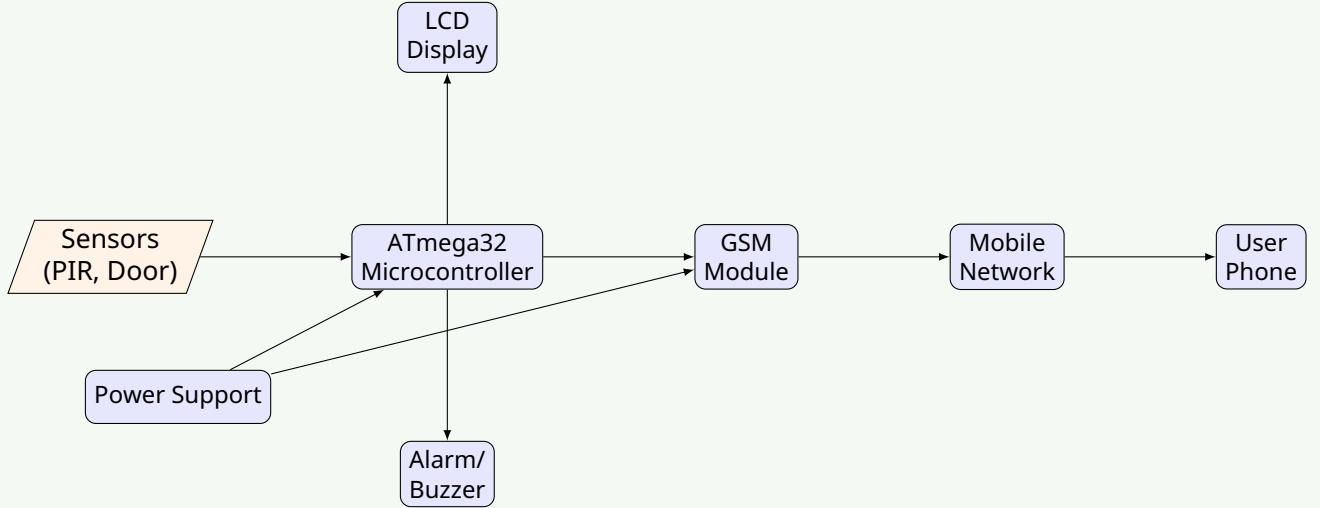
પ્રશ્ન 5(અ) [3 ગુણ]

GSM આધારિત સુરક્ષા સિસ્ટમનો બ્લોક ડાયાગ્રામ સમજાવો.

જવાબ

GSM સુરક્ષા સિસ્ટમ બ્લોક ડાયાગ્રામ:

આકૃતિ 6. GSM સુરક્ષા સિસ્ટમ



સિસ્ટમ ઘટકો:

કોષ્ટક 23. સિસ્ટમ ઘટકો

ઘટક	કાર્ય
સેન્સર્સ	PIR, ડોર/વિન્ડો સેન્સર્સ, સ્મોક ડિટેક્ટર
માઇક્રોકન્ટ્રોલર	સેન્સર ડેટા પ્રોસેસ કરે છે, સિસ્ટમ કન્ટ્રોલ
GSM મોડ્યુલ	SMS ચેતવણીઓ મોકલે છે, કોલ કરે છે
ડિસ્પ્લે	સિસ્ટમ સ્ટેટસ બતાવે છે
એલાર્મ	સ્થાનિક ઓડિયો/વિઝ્યુઅલ ચેતવણી

મેમરી ટ્રીક

“SGMA Security: Sensors, GSM module, Microcontroller, Alerts”

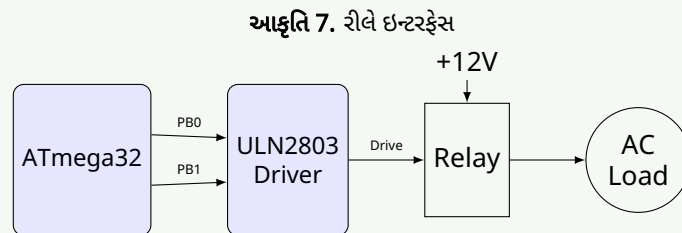
OR

પ્રશ્ન 5(બ) [4 ગુણ]

રીલે ઇન્ટરફેસ સમજાવો.

જવાબ

રીલે ઇન્ટરફેસ સર્કિટ:



રીલે ઇન્ટરફેસ કોડ:

```

1  #include <avr/io.h>
2  #include <util/delay.h>
3
4  void relay_init() {
5      DDRB |= (1 << PB0) | (1 << PB1); // Set as output pins
6  }
7
8  void relay1_on() {
9      PORTB |= (1 << PB0); // Activate relay 1
10 }
11
12 void relay1_off() {
13     PORTB &= ~(1 << PB0); // Deactivate relay 1
14 }
15
16 int main() {
17     relay_init();
18
19     while(1) {
20         relay1_on();    // Turn on relay 1
21         _delay_ms(2000);
22         relay1_off();   // Turn off relay 1
23         _delay_ms(1000);
24     }
25     return 0;
26 }
  
```

ULN2803 વિશેષતાઓ:

- 8 ચેનલો: આઠ ડાર્લિંગટન જોડી ડ્રાઇવરો
- ઉચ્ચ કરંટ: 500mA પ્રતિ ચેનલ સુધી
- સુરક્ષા: બિલ્ટ-ઇન ફ્લાયબેક ડાયોડ્સ

મેમરી ટ્રીક

“ULN Relay: ULN2803 driver, Load control, Non-contact switching”

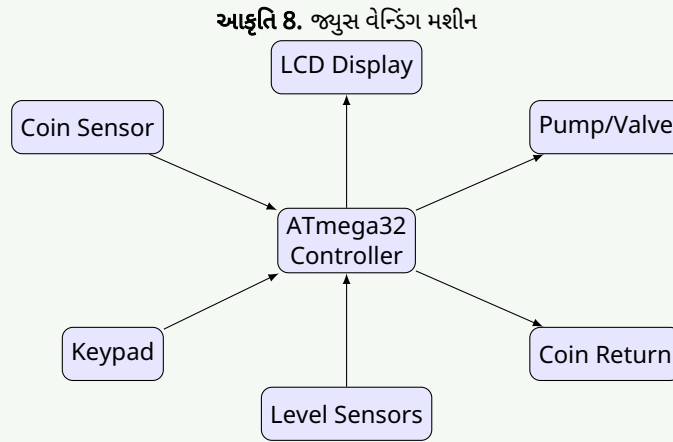
OR

પ્રશ્ન 5(ક) [7 ગુણ]

ઓટોમેટિક જ્યુસ વેન્ડિંગ મશીન દોરો અને સમજાવો

જવાબ

ઓટોમેટિક જ્યુસ વેન્ડિંગ મશીન બ્લોક ડાયાગ્રામ:



સિસ્ટમ ઓપરેશન:

- પ્રારંભિકીકરણ: વેલકમ મેસેજ અને જ્યુસ મેનુ દર્શાવો
- કોઇન ઇનપુટ: વપરાશકર્તા સિક્કા દાખલ કરે છે, સિસ્ટમ રકમ માન્ય કરે છે
- પસંદગી: વપરાશકર્તા જ્યુસ પ્રકાર પસંદ કરવા માટે કીપેડ દબાવે છે
- માન્યતા: તપાસો કે પૂરતા પૈસા અને જ્યુસ ઉપલબ્ધ છે કે કેમ
- ડિસ્પેન્સિંગ: પસંદ કરેલ જ્યુસ માટે પંપ અને વાલ્વ સક્રિય કરો
- પૂર્ણતા: જો કોઈ ફેરફાર હોય તો પરત કરો, આભાર સંદેશ દર્શાવો

કન્ટ્રોલ લોજિક:

```

1 // Pseudo code for vending machine operation
2 void vending_machine() {
3     display_menu();
4
5     while(1) {
6         if(coin_inserted()) {
7             total_amount += validate_coin();
8             update_display();
9         }
10
11         if(selection_made()) {
12             juice_type = get_selection();
13             if(total_amount >= juice_price[juice_type]) {
14                 if(juice_available[juice_type]) {
15                     dispense_juice(juice_type);
16                     return_change();
17                     reset_system();
18                 } else {
19                     display_error("Out of Stock");
20                 }
19             }
20         }
21     }
22 }
  
```

```
21     } else {  
22         display_error("Insufficient Amount");  
23     }  
24 }  
25 }  
26 }
```

મેમરી ટ્રીક

``CLPDV Juice: Coin sensor, LCD display, Pump motors, Dispensing unit, Valve control''