

마이크로프로세서 설계실험

TERM PROJECT: 결과 보고서



담당교수 : 이상문 교수님

2조 2015115304 이창현

2016116232 김신혁

제출일자: 2021년 6월 10일(목요일)

<목 차>

1. 목표

2. 프로젝트 설명 및 역할분담

3. 하드웨어 설계

4. 소프트웨어 설계

5. 실행 결과

6. 결론

7. 참조

8. 코드 리스트

1. 목표

NXP BOARD를 사용하여 개인용 전자금고를 만들고 실제 금고와 같은 기능을 구현해 본다.

2. 프로젝트 설명 및 역할 분담

이창현	코드 설계 및 구성, 발표, 보고서 작성
김신혁	하드웨어 설계 및 구성, 보고서 작성

저희가 구현하고자 하는 기능은 총 3가지가 존재합니다.

첫째, 입금 모드입니다. 입금 모드는 대기모드에 있는 NXP Board를 Push Button의 입력으로 동작하는 Interrupt를 이용하여 대기상태에서 벗어난 후 비밀번호를 입력하고 사용하고자 하는 모드 즉, 입금모드에 대한 커맨드('q')를 입력한 후 입력이 완료되었다는 커맨드('r')를 입력함으로써 입금모드로 진입하게 됩니다.

둘째, 출금 모드입니다. 출금 모드 또한 대기상태에 있는 NXP Board를 Push Button의 입력으로 동작하는 Interrupt를 이용하여 대기상태에서 벗어난 후 비밀번호를 입력하고 사용하고자 하는 모드 즉, 출금모드에 대한 커맨드('w')를 입력한 후 입력이 완료되었다는 커맨드('r')를 입력함으로써 출금모드로 진입하게 됩니다.

셋째, 암호 갱신 모드입니다. 암호갱신 모드 또한 대기상태에 있는 NXP Board를 Push Button의 입력으로 동작하는 Interrupt를 이용하여 대기상태에서 벗어난 후 비밀번호를 입력하고 사용하고자 하는 모드 즉, 암호갱신모드에 대한 커맨드('e')를 입력한 후 입력이 완료되었다는 커맨드('r')를 입력함으로써 암호갱신모드로 진입하게 됩니다.

입금모드로 진입한 NXP Board는 KeyBoard의 숫자키 패드를 통해 원하는 금액을 입력하게 됩니다. 입금하고자 하는 금액의 액수와 금고내에 저장되어 있는 기존의 금액의 합이 9999를 초과할 시 7-segment를 통해 저장할 수 있는 금액을 초과했다는 표시의 FAIL을 표시하게 됩니다. 이외의 상황에서는 기존의 금액과 입력한 금액의 액수를 더하여 7-segment에 표시됩니다.

출금모드에 진입한 NXP Board는 입력모드와 마찬가지로 숫자키 패드를 통해 원하는 금액을 입력하게 됩니다. 출금하고자 하는 금액이 금고내에 저장되어 있는 기존의 금액을 초과할 때 7-segment를 통해 금액이 부족하다는 것을 알려주기 위해 FAIL을 표시하게 됩니다. 이외의 상황에서는 기존의 금액에서 입력한 금액만큼 차감하여 7-segment에 표시됩니다.

암호갱신모드에 진입한 NXP Board는 4자리 숫자의 새로운 암호를 입력으로 받고, 해당 암호를 기존의 암호와 교체하여 다시금 금고를 열기 위해서는 새로운 암호를 입력하여야 합니다. 7-segment에서는 암호갱신이 성공적으로 이뤄졌다는 것을 알리는 SUCCESS가 7-segment에 표시됩니다.

마지막으로 암호를 입력하는 과정에서 특정 모드를 사용하겠다는 커맨드를 입력하지 않을 시에는 모드를 입력하지 않았기 때문에 잘못된 접근을 막기위해 FAIL을 7-segment에 표시됩니다.

【그림1: 7-Segment 문자 테이블】

위의 출력은 숫자와 관련된 모든 입력은 Keyboard로 입력하고 I/O Cable을 통해 NXP에 입력하여 계산되면 7-Segment에 이루어지도록 합니다.

0	0	A	A	K	E	U	U
1	1	B	b	L	L	V	8
2	2	C	c	M	A	W	2
3	3	D	d	N	n	X	11
4	4	E	E	O	O	Y	3
5	5	F	F	P	P	Z	3
6	6	G	G	Q	9		
7	7	H	H	R	r		
8	8	I	1	S	5		
9	9	J	J	T	t		

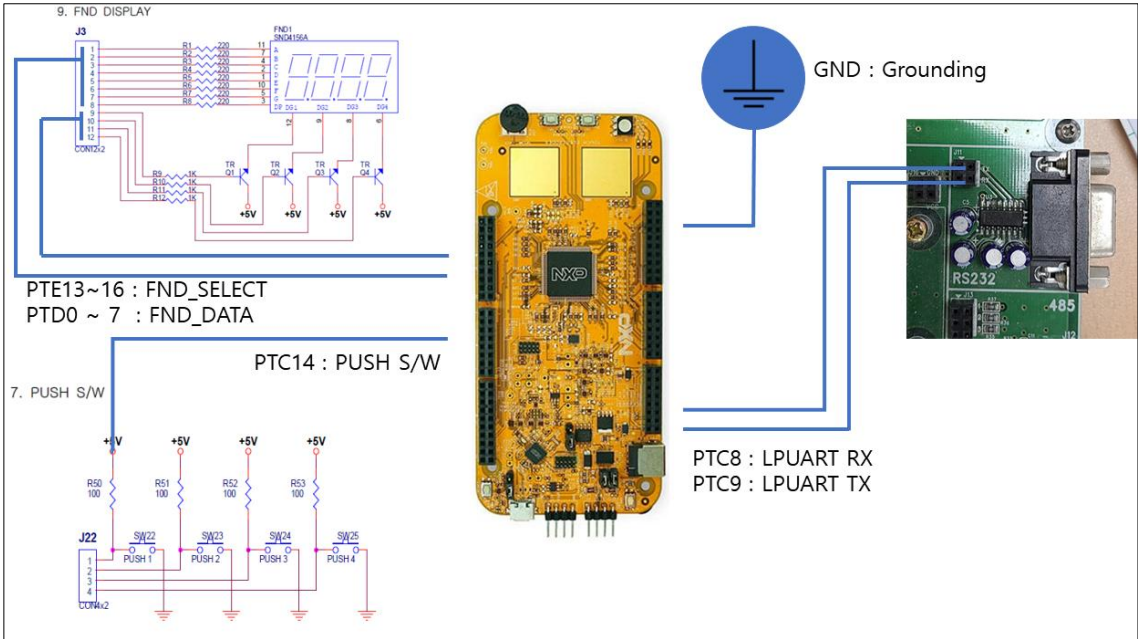
이 외에도 여러 가지 모드가 있습니다. 첫째는 대기 모드입니다. 대기 중에는 7-Segment에 Wait가 표시되도록 하여 입력 대기 중임을 나타냅니다. 두 번째는 성공 출력 모드입니다. 입/출금을 완료하거나 비밀번호를 성공적으로 갱신 시, Success를 출력합니다. 한번에 표현할 수 있는 글자수는 4글자이므로 전광판과 같이 우측에서 좌측으로 지나가는

글자를 구현합니다. 세 번째는 실패 출력 모드입니다. 입금을 표현할 수 있는 수치 이상으로 입금하거나 보유 자산을 초과하는 금액을 출금하려 할 때 진행 불가능함을 표현하기 위해 FAIL을 출력합니다. 네 번째는 오픈 모드입니다. 올바른 암호를 입력 시 금고가 열렸음을 표현하기 위해 OPEN을 7-Segment에 출력합니다.

또한 7-Segment에서는 알파벳을 온전한 모양으로 출력하기 어렵기 때문에 우측의 문자 테이블에 따라 표기합니다.

3. 하드웨어 설계

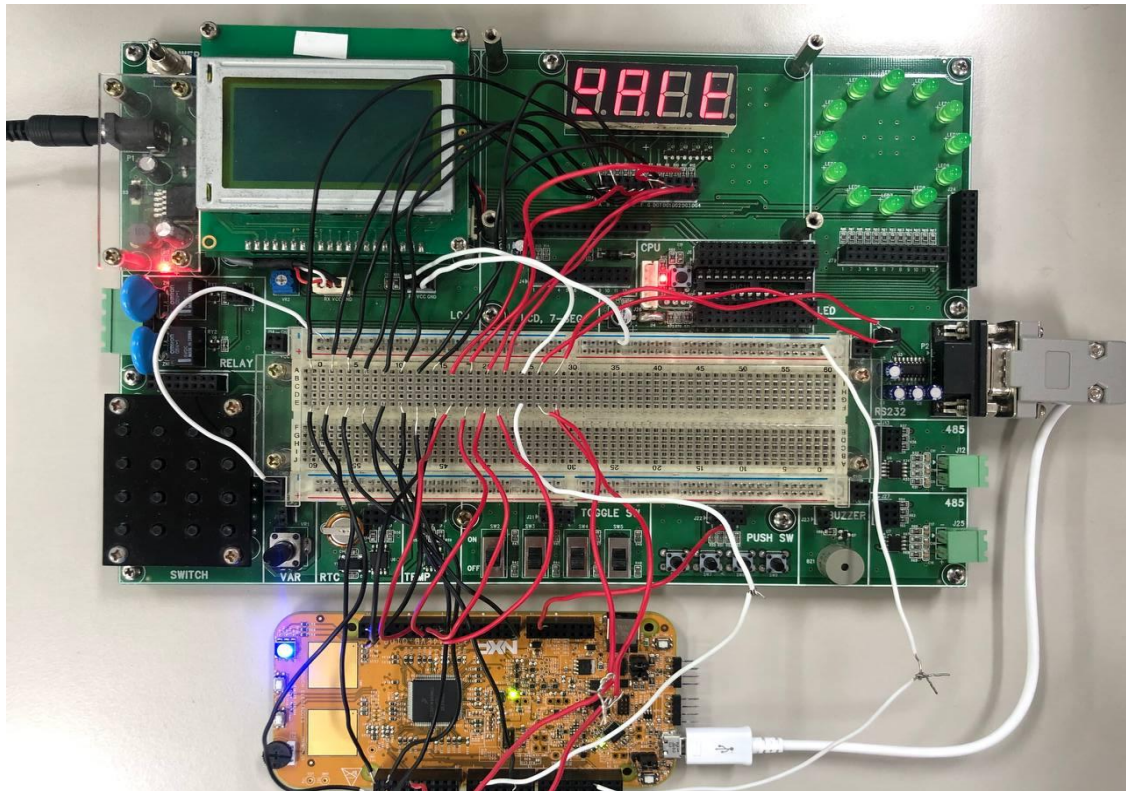
[그림 2: 하드웨어 결선도]



Port	Hardware	Port	Hardware
PTC8	LPUART RX	PTE13 ~ PTE16	FND_Select
PTC9	LPUART TX	PTD0 ~ PTD7	FND_Data
PTC14	Push Button	GND	grounding

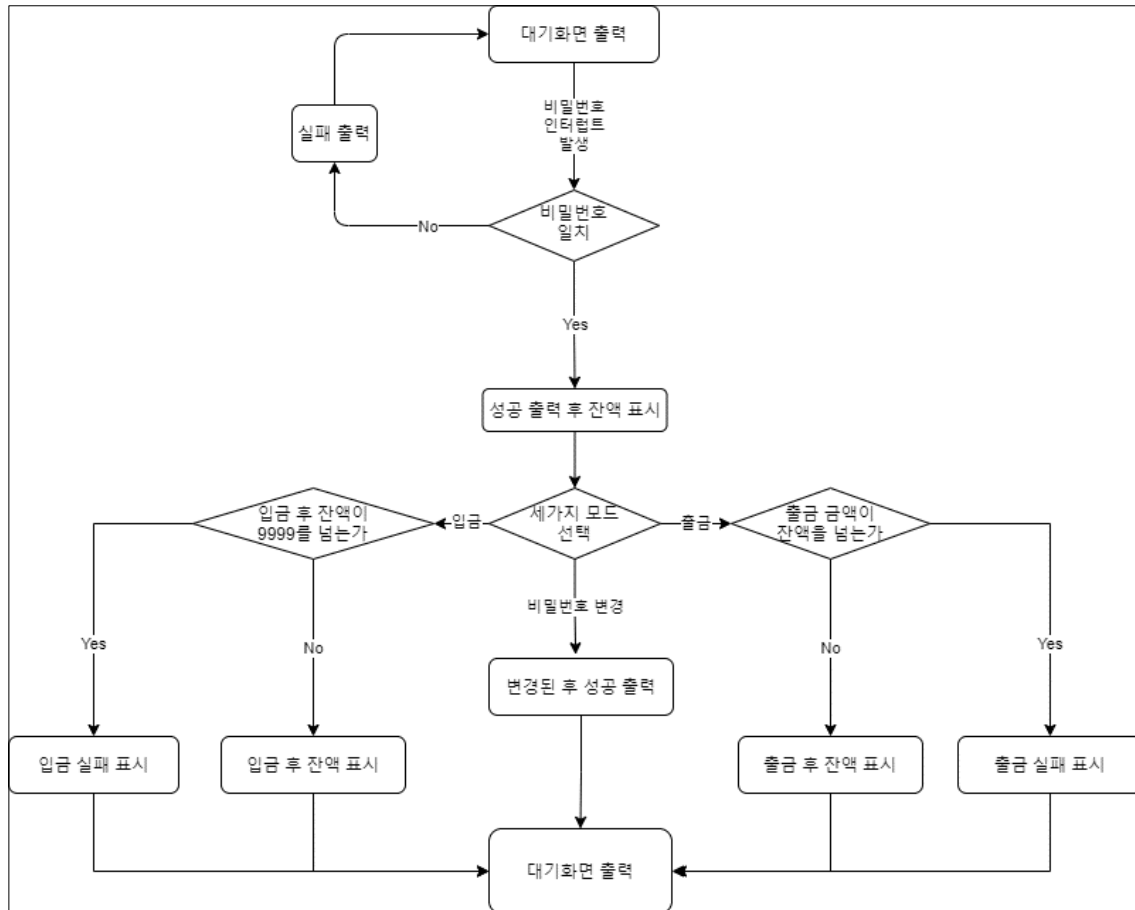
장비	역할
7-Segment	출력하고자 하는 수치를 표시한다.
Push button	대기모드로 진입하거나, 대기모드에서 벗어날 수 있게 합니다.
I/O Cable	Keyboard의 입력을 NXP Board에 입력할 수 있도록 연결한다.
Keyboard	원하는 숫자와 모드 진입키를 입력한다.
Breadboard	여러 장치들을 결선하기 위해 사용합니다.

[그림 3: 하드웨어 구현]



4. 소프트웨어 설계

[그림 4: 플로우 차트]



큰 맥락으로 7가지 함수를 구현할 계획입니다.

첫 번째 함수는 메인 함수입니다. 메인 함수에서는 입력이 들어오기 전까지 대기 모드를 작동하며 화면에는 지정된 대기 화면을 출력합니다. 그러다가 입력이 생기면 다음 진행할 함수를 호출합니다.

두 번째 함수는 use_uart 함수입니다. 이 함수는 하이퍼터미널에 키보드로 입력을 받아 지정된 변수에 저장해 수치 혹은 모드를 결정하는 함수입니다.

세 번째 함수는 FND_select 함수입니다. 이 함수는 FND에 나타낼 수치를 받는 함수입니다.

네 번째 함수는 fnd_sucess 함수입니다. 성공적으로 처리되었음을 SUCCESS로 나타내는 함수입니다. 전광판처럼 우측에서 좌측으로 글자가 지나갑니다.

다섯 번째 함수는 fnd_wait 함수입니다. 입력 대기 중임을 FND로 나타내는 함수입니다.



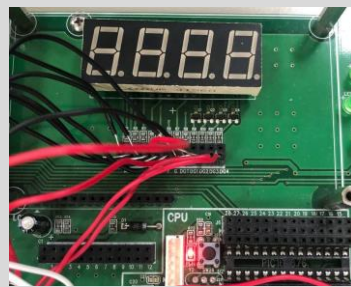
여섯 번째 함수는 fnd_open 함수입니다. 올바른 비밀번호 입력 시 금고가 열렸음을 FND에

OPEN으로 출력하는 함수입니다.

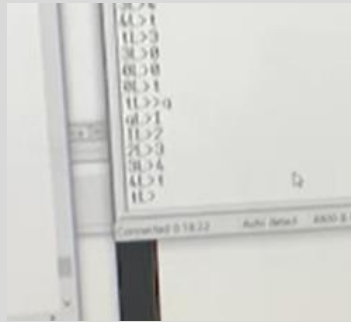


일곱 번째 함수는 fnd_fail 함수입니다. 잘못된 값이 들어왔을 때 실패했음을 FND에 Fail로 출력하는 함수입니다.

5. 실행 결과




1) 대기 화면

<p>그림 5: WAIT를 출력하고 있다.</p> 	<p>그림 6: 대기 모드를 종료하는 버튼을 누르는 모습이다.</p> 	<p>그림 7: 값을 입력받기 위해 FND가 꺼진 모습이다.</p> 
<p>대기 중에는 성공적으로 'WAIT'를 출력하는 모습입니다.</p>	<p>대기 모드를 종료하기 위해 버튼을 눌렀습니다.</p>	<p>성공적으로 대기 모드가 종료된 모습입니다.</p>

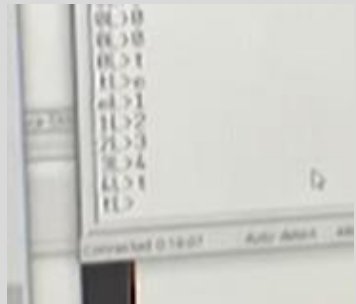
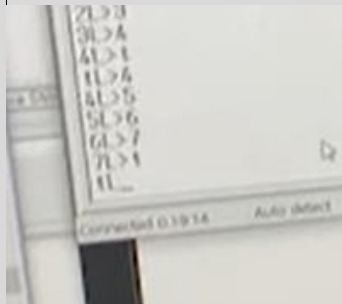

2) 입금

<p>그림 8: q 1 2 3 4 t 를 입력하였다.</p> 	<p>그림 9: 3 0 0 t를 입력하였다.</p> 	<p>그림 10: 5300이 표시되는 모습이다.</p> 
<p>입금 모드 진입 키인 q를 입력 후 올바른 암호 '1234'를 입력한 뒤 완료커맨드 't'를 입력하였습니다.</p>	<p>그 후 입금할 금액 300을 입력 후 't'를 입력하였습니다.</p>	<p>기존 자산인 5000에 300이 더해져 5300이 성공적으로 뜬 모습입니다.</p>

3) 출금

<p>그림 11: w 1 2 3 4 t를 입력한 모습이다.</p> 	<p>그림 12: 5 0 0 0 t를 입력한 모습이다.</p> 	<p>그림 13: 0300이 표시되고 있다.</p> 
<p>출금 모드 진입키인 'w'를 입력한 후 올바른 암호 1234를 넣고 완료커맨드 't'를 입력한 모습입니다.</p>	<p>출금 금액인 5000을 입력한 후 't'를 입력한 모습입니다.</p>	<p>이전 자산 5300에서 5000이 감소되어 300이 성공적으로 출력되는 모습입니다.</p>

4) 암호 갱신

<p>그림 14: e 1 2 3 4 t를 입력한 모습이다.</p> 	<p>그림 15: 4 5 6 7 t를 입력한 모습이다.</p> 	<p>그림 16: Success에서 e가 잠깐 표시되는 중이다.</p> 
<p>암호 갱신 모드 커맨드 'e'를 입력 후 올바른 비밀번호 1234를 넣고 t를 입력한 모습입니다.</p>	<p>새로 갱신할 비밀번호 4567을 입력 후 t를 입력한 모습입니다.</p>	<p>성공적으로 바뀌었음을 나타내는 Success가 출력되어 지나가는 모습입니다.</p>

5) 실패 화면

그림 17: FAIL이 표시되는 모습이다.



비밀번호를 틀리거나 입금 한도를 초과하는 경우 등 실패가 터야 하는 상황에서 FAIL이 정상적으로 출력되는 모습입니다.

6. 결론

기존 보유액인 5000에서 입출금액에 따라 정상적으로 감가잔액이 계산되어 출력되며, 동시에 사전에 설정된 비밀번호와 일치 할 경우 동작을 수행하고 일치 하지 않은 경우에는 FAIL을 출력하며 동작을 멈추게 됩니다.

현재 문제점은 표시할 수 있는 자릿수가 네 자릿수에 그쳤다는 것이 있으며, 하이퍼 터미널과의 연결이 불안정해 똑같은 환경에서도 노이즈가 발생하는 경우 정상적인 작동을 기대할 수 없다는 점이 있습니다.

또한 첫 번째 문제에 대한 해결법으로 4x4 키패드를 사용하는 방법이 제시될 수 있습니다. 하지만 저희가 초기에 입력을 받기위해 4x4 Keypad를 사용하고자 하였으나 실제로 구현하였을 때 키패드가 입력될 때 하드웨어적인 문제를 발견하였습니다. Keypad를 한번 입력하였으나 여러 번 입력되거나, 입력조차 되지 않는 경우가 생겼습니다. 그렇기에 저희는 UART통신을 이용하여 keyboard에서 입력을 받아오게 되었습니다.

두 번째 문제로는 해당코드를 하이퍼터미널을 통해 실행시키면 초기에 Auto detcet로 작동하는데 조금의 시간이 지나면 설정한 설정값들로 '4800 8-N-1'으로 통신이 제대로 이뤄지는 문제를 발견하였습니다.

7. 참조

- 7-Segment 문자 테이블 : <https://mul-ku.tistory.com/entry/7SEGMENT-문자표>
- 하드웨어 결선도에 사용한 그림 : 강의자료 3주차_마이크로프로세서 설계실험_v7.pdf

8. 코드 리스트

```
1. #include "device_registers.h"
2. #include "clocks_and_modes.h"
3. #include "LPUART.h"
4. #include <stdio.h>
5. #include <stdlib.h> //인터럽트, UART를 사용하기 위한 헤더파일
6.
7. unsigned int FND_DATA[10]={0x3F, 0x06, 0x5b, 0x4F, 0x66, 0x6D, 0x7C, 0x07, 0x7F, 0x67};
8. unsigned int FND_SEL[5]={0x2000,0x4000,0x8000,0x10000};
9. unsigned int FND_ENG[13]={0b01110001,0b01110111,0b00110000, 0b00111000,
    0b01011100,0b01110011,0b01111001, 0b01010101, 0b01011110, 0b01101010,
    0b01111000,0b01010100}; //순서대로 알파벳 f a l l o p e m d w t N을 FND에 나타낼 수
    있도록 지정한 것입니다.
10.
11. unsigned int num=0,num0=0,num1=0,num2=0,num3=0;
12. unsigned int Dtime= 0;
13. unsigned int uart1 = 0; //인터럽트를 이용한 대기모드 조작에 사용됩니다.
14.
15. int i=0;
16. char data = 0;
17. char transfer_data[20];
18. int j=0;
19. int lpit0_ch0_flag_counter = 0;
20.
21. unsigned int mode = 0;
22. unsigned int pw = 1234; //기본 비밀번호는 1234입니다.
23. unsigned int pw1 = 0;
24. unsigned int pw2 = 0;
25. unsigned int number = 0;
26. unsigned int money = 5000; //기존 금액은 5000입니다.
```

```

27. unsigned int wait = 0; //대기상태를 나타내는 변수입니다.
28.
29. int num4 = 0; //반복적인 인터럽트 구문을 사용하기위해서 선언하였습니다.
30.
31. void PORT_init(void) //하드웨어에서 사용된 PORT들을 초기화하였습니다.
32. {
33.     PCC->PCCn[PCC_PORTA_INDEX]=PCC_PCCn_CGC_MASK;
34.     PCC->PCCn[PCC_PORTC_INDEX]=PCC_PCCn_CGC_MASK;
35.     PCC->PCCn[PCC_PORTD_INDEX]=PCC_PCCn_CGC_MASK;
36.     PCC->PCCn[PCC_PORTE_INDEX]=PCC_PCCn_CGC_MASK;
37.
38.     PTD->PDDR |= 1<<0|1<<1|1<<2|1<<3|1<<4|1<<5|1<<6|1<<7;
39.     PTE->PDDR |= 1<<13|1<<14|1<<15|1<<16;
40.     PTC->PDDR &= ~(1<<4);
41.     PORTC->PCR[4] |= PORT_PCR_MUX(1);
42.     PORTC->PCR[4] |= (10<<16);
43.
44.     PORTC->PCR[8] |= PORT_PCR_MUX(2);
45.     PORTC->PCR[9] |= PORT_PCR_MUX(2);
46.
47.     PORTA->PCR[3] |= PORT_PCR_MUX(6);
48.
49.     PORTD->PCR[0] = PORT_PCR_MUX(1);/* PortD0: MUX = GPIO */
50.     PORTD->PCR[1] = PORT_PCR_MUX(1);/* PortD1: MUX = GPIO */
51.     PORTD->PCR[2] = PORT_PCR_MUX(1);/* PortD2: MUX = GPIO */
52.     PORTD->PCR[3] = PORT_PCR_MUX(1);/* PortD3: MUX = GPIO */
53.     PORTD->PCR[4] = PORT_PCR_MUX(1);/* PortD4: MUX = GPIO */
54.     PORTD->PCR[5] = PORT_PCR_MUX(1);/* PortD5: MUX = GPIO */
55.     PORTD->PCR[6] = PORT_PCR_MUX(1);/* PortD6: MUX = GPIO */
56.     PORTD->PCR[7] = PORT_PCR_MUX(1);/* PortD7: MUX = GPIO */
57.

```

```

58.         PORTE->PCR[13] = PORT_PCR_MUX(1); /*PortD13: mux= GPIO */
59.         PORTE->PCR[14] = PORT_PCR_MUX(1); /*PortD14: mux= GPIO */
60.         PORTE->PCR[15] = PORT_PCR_MUX(1); /*PortD15: mux= GPIO */
61.         PORTE->PCR[16] = PORT_PCR_MUX(1); /*PortD16: mux= GPIO */
62. }
63.
64. void WDOG_disable(void)
65. {
66.     WDOG->CNT=0xD928C520;
67.     WDOG->TOVAL=0x0000FFFF;
68.     WDOG->CS = 0x00002100;
69. }
70.
71. void LPIT0_init(uint32_t delay)
72. {
73.     uint32_t timeout;
74.     PCC->PCCn[PCC_LPIT_INDEX] = PCC_PCCn_PCS(6);
75.     PCC->PCCn[PCC_LPIT_INDEX] |= PCC_PCCn_CGC_MASK;
76.
77.     LPIT0 -> MCR |= LPIT_MCR_M_GEN_MASK;
78.
79.     timeout = delay * 40000;
80.     LPIT0->TMR[0].TVAL = timeout;
81.     LPIT0->TMR[0].TCTRL |= LPIT_TMR_TCTRL_T_EN_MASK;
82. }
83.
84. void delay_ms (volatile int ms)
85. {
86.     LPIT0_init(ms);
87.     while(0 == (LPIT0->MSR & LPIT_MSR_TIFO_MASK)){
88.         lpit0_ch0_flag_counter++;

```

```

89.         LPIT0->MSR |= LPIT_MSR_TIFO_MASK;
90. }
91.
92. void NVIC_init_IRQs(void){
93.     S32_NVIC->ICPR[1] |= 1<<(61%32);
          //PORT C를 Interrupt로 사용하기 위해 설정하였습니다.
94.     S32_NVIC->ISER[1] |= 1<<(61%32);
95.     S32_NVIC->IP[61] =0xB;
96. }
97.
98. void PORTC_IRQHandler(void)
99. {
100.     PORTC->PCR[4] &= ~(0x01000000);
101.
102.     if(((PORTC->ISFR&(1<<4)) != 0))
          //인터럽트 발생 시 상태변수 wait를 반복적으로 바꿔줍니다.
103.     {
104.         delay_ms(10);
105.         num4 += 1;
106.
107.         if(num4 % 2 == 1)
108.         {
109.             wait = 1;
110.         }
111.
112.         else if(num4 % 2 == 0)
113.         {
114.             wait = 0;
115.         }
116.     }
117.     PORTC->PCR[4] |= 0x01000000; //인터럽트 초기화

```



```

118. }
119.
120. void LPUART0_init(void)
121. {
122.     PCC->PCCn[PCC_LPUART0_INDEX] &= ~PCC_PCCn_CGC_MASK;
123.     PCC->PCCn[PCC_LPUART0_INDEX] |= PCC_PCCn_PCS(0x001) | PCC_PCCn_CGC_MASK;
124.
125.     LPUART0->BAUD = 0x0F000068; //4800Hz를 설정하였습니다.
126.
127.     LPUART0->CTRL = 0x000C0000;
128. }
129.
130. void LPUART0_transmit_char(char send)
131. {
132.     while((LPUART0->STAT & LPUART_STAT_TDRE_MASK)>>LPUART_STAT_TDRE_SHIFT==0);
133.
134.     LPUART0->DATA=send;
135. }
136.
137. void LPUART0_transmit_string(char data_string[])
138. {
139.     uint32_t i=0;
140.     while(data_string[i] != '\0')
141.     {
142.         LPUART0_transmit_char(data_string[i]);
143.         i++;
144.     }
145. }
146.
147. char LPUART0_receive_char(void)
148. {

```

```

149. static char receive;
150. while((LPUART0->STAT & LPUART_STAT_RDRF_MASK)>>LPUART_STAT_RDRF_SHIFT==0);
151.
152. receive = LPUART0 -> DATA;
153. return receive;
154. }
155.
156. void fnd_fail()//잘못된 값이 들어왔을 때 실패했음을 FND에 Fail로 출력하는 함수입니
    다.
157. {
158.     PTE->PSOR = FND_SEL[j];
159.     PTD->PCOR = FND_ENG[0];
160.     delay_ms(Dtime/300);
161.     PTD->PSOR =0xff;
162.     PTE->PCOR = 0xfffff;
163.     j++;
164.
165.     PTE->PSOR = FND_SEL[j];
166.     PTD->PCOR = FND_ENG[1];
167.     delay_ms(Dtime/300);
168.     PTD->PSOR =0xff;
169.     PTE->PCOR = 0xfffff;
170.     j++;
171.
172.     PTE->PSOR = FND_SEL[j];
173.     PTD->PCOR = FND_ENG[2];
174.     delay_ms(Dtime/300);
175.     PTD->PSOR =0xff;
176.     PTE->PCOR = 0xfffff;
177.     j++;
178.

```

```

179.    PTE->PSOR = FND_SEL[j];
180.    PTD->PCOR = FND_ENG[3];
181.    delay_ms(Dtime/300);
182.    PTD->PSOR =0xff;
183.    PTE->PCOR = 0xfffff;
184.    j=0;
185. }
186.
187. void fnd_open()
188. {
    //올바른 비밀번호 입력 시 금고가 열렸음을 FND에 OPEN으로 출력하는 함수입니다.
189.    PTE->PSOR = FND_SEL[j];
190.    PTD->PCOR = FND_ENG[4];
191.    delay_ms(Dtime/300);
192.    PTD->PSOR =0xff;
193.    PTE->PCOR = 0xfffff;
194.    j++;
195.
196.    PTE->PSOR = FND_SEL[j];
197.    PTD->PCOR = FND_ENG[5];
198.    delay_ms(Dtime/300);
199.    PTD->PSOR =0xff;
200.    PTE->PCOR = 0xfffff;
201.    j++;
202.
203.    PTE->PSOR = FND_SEL[j];
204.    PTD->PCOR = FND_ENG[6];
205.    delay_ms(Dtime/300);
206.    PTD->PSOR =0xff;
207.    PTE->PCOR = 0xfffff;
208.    j++;

```

```

209.
210.   PTE->PSOR = FND_SEL[j];
211.   PTD->PCOR = FND_ENG[11];
212.   delay_ms(Dtime/300);
213.   PTD->PSOR =0xff;
214.   PTE->PCOR = 0xfffff;
215.   j=0;
216. }
217. void fnd_wait()
218. {
    //입력 대기 중임을 FND로 나타내는 함수입니다.
219.   PTE->PSOR = FND_SEL[j];
220.   PTD->PCOR = FND_ENG[9];
221.   delay_ms(Dtime/300);
222.   PTD->PSOR =0xff;
223.   PTE->PCOR = 0xfffff;
224.   j++;
225.
226.   PTE->PSOR = FND_SEL[j];
227.   PTD->PCOR = FND_ENG[1];
228.   delay_ms(Dtime/300);
229.   PTD->PSOR =0xff;
230.   PTE->PCOR = 0xfffff;
231.   j++;
232.
233.   PTE->PSOR = FND_SEL[j];
234.   PTD->PCOR = FND_ENG[2];
235.   delay_ms(Dtime/300);
236.   PTD->PSOR =0xff;
237.   PTE->PCOR = 0xfffff;
238.   j++;

```

```

239.
240.   PTE->PSOR = FND_SEL[j];
241.   PTD->PCOR = FND_ENG[10];
242.   delay_ms(Dtime/300);
243.   PTD->PSOR =0xff;
244.   PTE->PCOR = 0xfffff;
245.   j=0;
246. }
247. void fnd_sucess()
    //성공적으로 처리되었음을 SUCCESS로 나타내는 함수입니다. 전광판처럼 우측에서 좌측
    으로 글자가 지나갑니다.
248. {
249.   unsigned int  FND_SUCDATA[15]={0b00000000, 0b00000000, 0b00000000,
    0b00000000, 0b01101101, 0b00011100, 0b01011000, 0b01011000, 0b01111001,
    0b01101101, 0b01101101, 0b00000000, 0b00000000, 0b00000000, 0b00000000};
    //차례대로 null null null null s u c c e s s null null null null입니다.
250.   for(int x = 1; x < 12; x++){
251.       for(int y = 0; y<4 ; y++){
252.           PTE->PSOR = FND_SEL[y];
253.           PTD->PCOR = FND_SUCDATA[x+y];
254.           delay_ms(100);
255.           PTD->PSOR =0xff;
256.           PTE->PCOR = 0xfffff;
257.       }
258.   }
259. }
260.
261. void FND_select(int num) //FND에 나타낼 수를 받는 함수입니다.
262. {
263.   num3=(num/1000) % 10;
264.   num2=(num/100) % 10;

```

```
265.    num1=(num/10) % 10;
266.    num0= num % 10;
267.
268.    PTE->PSOR = FND_SEL[j];
269.    PTD->PCOR = FND_DATA[num3];
270.    delay_ms(Dtime/300);
271.    PTD->PSOR =0xff;
272.    PTE->PCOR = 0xfffff;
273.    j++;
274.
275.    PTE->PSOR = FND_SEL[j];
276.    PTD->PCOR = FND_DATA[num2];
277.    delay_ms(Dtime/300);
278.    PTD->PSOR =0xff;
279.    PTE->PCOR = 0xfffff;
280.    j++;
281.
282.    PTE->PSOR = FND_SEL[j];
283.    PTD->PCOR = FND_DATA[num1];
284.    delay_ms(Dtime/300);
285.    PTD->PSOR =0xff;
286.    PTE->PCOR = 0xfffff;
287.    j++;
288.
289.    PTE->PSOR = FND_SEL[j];
290.    PTD->PCOR = FND_DATA[num0];
291.    delay_ms(Dtime/300);
292.    PTD->PSOR =0xff;
293.    PTE->PCOR = 0xfffff;
294.    j=0;
295. }
```

296.

297. void use_uart(void)

//하이퍼터미널에 키보드로 입력을 지정된 변수에 받아 수치 혹은 모드를 결정하는
함수입니다.

298. {

299. char msg_init = 0x00;

300. char msg_clear = 0xA3;

301. char msg_reset = 0xA0;

302. char msg_Start = 0xA2;

303.

304. char buffer = 0;

305. char array_data[20]="";

306.

307. LPUART1_transmit_char('>');

308.

309. buffer=LPUART1_receive_and_echo_char();

310. switch (buffer) //키보드의 숫자키를 이용하여 입력값을 받아옵니다.

311. {

312. case '0':

313. number = (number*10);

314. break;

315.

316. case '1':

317. number = (number*10)+ 1;

318. break;

319.

320. case '2':

321. number = (number*10)+ 2;

322. break;

323.

324. case '3':

```
325.                 number = (number*10)+ 3;
326.         break;
327.
328.         case '4':
329.                 number = (number*10)+ 4;
330.         break;
331.
332.         case '5':
333.                 number = (number*10)+ 5;
334.         break;
335.
336.         case '6':
337.                 number = (number*10)+ 6;
338.         break;
339.
340.         case '7':
341.                 number = (number*10)+ 7;
342.         break;
343.
344.         case '8':
345.                 number = (number*10)+ 8;
346.         break;
347.
348.         case '9':
349.                 number = (number*10)+ 9;
350.         break;
351.
352.         case 'q': //입금모드
353.         mode = 1;
```



```

354.         break;
355.
356.         case 'w': //출금모드
357.             mode = 2;
358.             break;
359.
360.         case 'e': //암호갱신모드
361.             mode = 3;
362.
363.             break;
364.
365.         case 'r':
366.             uart1 = 1; //입력이 완료되었다는 표시를 하는 변수
367.
368.             break;
369.
370.         default:
371.             break;
372.     }
373.
374.     LPUART1_transmit_char(buffer);
375.     LPUART1_transmit_char("\n\r");
376. }
377.
378.
379. int main(void)
    //입력이 생기기 전까지 대기 중이었다가 입력이 생기면 그 다음으로 진행해주는
    맥락을 가진 함수입니다.
380. {
381.     WDOG_disable();
382.     SOSCS_init_8MHz();

```

```

383.     SPLL_init_160MHz();
384.     NormalRUNmode_80MHz();
385.     SystemCoreClockUpdate();
386.         NVIC_init_IRQs(); /*InterruptPending, Endable, PrioritySet*/
387.
388.     PORT_init();
389.     LPUART0_init();
390.     LPUART1_init();
391.     Dtime = 500; //기본적인 설정을 초기화해주는 함수입니다.
392.
393. for(;;) //연속적인 동작을 위한 무한루프
394. {
395.     for(;;) //인터럽트에 따른 대기상태를 유지하기 위한 무한루프
396.     {
397.         if(wait == 0) //대기상태
398.         {
399.             fnd_wait();
400.
401.             number = 0;
402.             uart1 = 0;
403.
404.         }
405.
406.         else if(wait == 1) //대기상태를 해제하고 입력을 받습니다.
407.         {
408.             use_uart();
409.             pw1 = number;
410.
411.         }
412.
413.         if(uart1 == 1) //입력이 완료되면 멈춥니다.
414.         {
415.             break;
416.         }
417.     }
418. }

```

```

414.
415.     switch(mode)
416.     {
417.         case 0: //모드가 입력되지 않았기 때문에 탈출합니다.
418.             uart=0;
419.             number=0;
420.             break;
421.         case 1: //입금모드
422.             number = 0;
423.             uart1 = 0;
424.             for(;;)
425.             {
426.                 if(pw != pw1) //비밀번호가 틀리면 탈출
427.                 {
428.                     break;
429.                 }
430.                 else if(pw == pw1)
431.                     //비밀번호 일치 시 입금금액 입력받음
432.                     {
433.                         use_uart();
434.
435.                     }
436.                     if(uart1 == 1)
437.                         //입력완료시 기존의 금액에 더한다.
438.                         {
439.                             money += number;
440.                             break;
441.                         }
442.                     }
443.             }
444.             break;

```

```

442.         case 2: //출금모드
443.             number = 0;
444.             uart1 = 0;
445.             for(;;)
446.             {
447.                 if(pw != pw1) //비밀번호가 틀리면 탈출
448.                 {
449.                     break;
450.                 }
451.                 else if(pw == pw1)
452.                     //비밀번호 일치 시 출금금액 입력받음
453.                     {
454.                         use_uart();
455.                     }
456.                 if(uart1 == 1)
457.                     //입력완료시 기존의 금액에 뺀다.
458.                     {
459.                         money -= number;
460.                         break;
461.                     }
462.                 break;
463.
464.         case 3: //비밀번호 갱신모드
465.             number = 0;
466.             uart1 = 0;
467.             for(;;)
468.             {
469.                 if(pw != pw1)

```

```

470.         //비밀번호 틀리면 탈출
471.         {
472.             break;
473.         }
474.         else if(pw == pw1)
475.         {
476.             //비밀번호 일치시 새로운 패스워드 입력받음
477.             use_uart();
478.
479.         }
480.         if(uart1 == 1)
481.         {
482.             pw2 = number;
483.             //새로운 비밀번호변수에 입력
484.             break;
485.         }
486.     }
487.     break;
488. default :
489.     break;
490. }
491. for(;;)
492. {
493.     if(pw != pw1) //비밀번호 틀리면 FAIL
494.     {
495.         fnd_fail();
496.     }
497.     else if(mode == 3) //비밀번호 바꾸고 SUCCESS출력
498.     {

```

```

496.                pw = pw2;
497.                fnd_sucess();
498.            }
499.            else if(money < 0) //잔액이 부족하면 FAIL 출력
500.            {
501.                fnd_fail();
502.            }
503.            else if(money > 9999)
504.                //잔액이 표시가능 금액 초과시 FAIL출력
505.            {
506.                fnd_fail();
507.            }
508.            else
509.                //적절한 입력이 수행되었을 때 잔액을 표시해줍니다.
510.            {
511.                FND_select(money);
512.            }
513.            if(wait == 0)
514.                //인터럽트가 발생하여 대기모드로 다시 진입 시에 사용됩니다.
515.            {
516.                break;
517.            }
518.        }

```