Embedded 1st HW. Calculator on Board with IPC
(설계 프로젝트 수행 결과)

과목명: [CSE4116] Embedded Systems
담당교수: 서강대학교 컴퓨터공학과 박 성 용

학번 및 이름: 20091648, 이준호
개발기간: 2014. 04. 08. - 2014. 04. 16.

최종 보고서

I. 개발 목표

본 프로젝트에서는 device control과 IPC를 이용하여 주어진 stop-watch와 text editor를 주어진 보드 위에서 작동하도록 구현한다.

II. 개발 범위 및 내용

자신들이 설계한 개발 목표를 달성하기 위하여 어떠한 내용의 개발을 수행할 지 그 범위와 개발 내용을 기술할 것.

가. 개발 범위

Input process - 이 프로세스는 계산을 위해 입력을 받는 입력 장치를 관리한다. 주기적으로 입력 장치로부터 새로 들어온 값이 있는지 확인하고, 새로 들어온 값을 main process로 IPC를 사용하여 전달한다. 입력 장치는 gpio의 7개의 스위치와 fpga의 9개의 스위치를 사용한다.

Main process - 이 프로세스에서는 연산의 실질적인 수행을 한다. 입력 프로세스에게서 값을 받아서, 각 모드에 따를 적절한 연산을 한다. 그 다음, 연산된 값을 IPC를 통해 출력 프로세스로 전달하는 역할을 한다.

Output process - 이 프로세스는 main process로부터 전달 받은 값을 해당하는 장치에 출력하는 역할을 한다. 모드 1에서는 gpio의 LED와 FND 출력, 모드 2에서는 fpga text lcd, dot matrix, fnd, led에 출력한다. 본 프로그램의 IPC는 shared-memory를 이용하여 구현하였다.

Device control - Device를 컨트롤하는 방법에는 mmap() 함수를 이용하는 방법과 device driver를 사용하는 방법이 있다. gpio의 출력은 mmap 함수를 사용하고, fpga의 출력은 device driver를 사용하여 device를 컨트롤한다. 이 두 가지 device 컨트롤을 모두 사용하여 프로그램을 구현해야 한다.

나. 개발 내용

Stop watch, text editor 기능을 구현한다. Mode 1에서는 gpio를 사용하여 stop watch 기능을, mode 2 에서는 fpga를 사용하여 text editor 기능을 구현한다. 기본적으로 mode가 변경되면, 각 mode의 초기 상태가 된다.

1) Stop watch

FND : 시간을 출력한다. (앞 두 자리는 분(60분), 뒤 두 자리는 초 (60초)). 60분이 넘어가면, 0으로 돌아가서 이어진다. 초기 상태는 0000.

LED : 초기 상태는 1번 LED에만 불이 들어온 상태, stop watch를 시작하면 3, 4번 LED에 불이 들어온다. Stop watch가 시간을 계속 카운트 하는 동안은 3, 4번 LED에 들어오고, pause를 누르면 3, 4번 LED를 1초에 한번씩 번갈아 가면서 불이 들어오게 한다. (한 LED를 1초씩 켰다 껏다를 반복) Stop watch의 reset 버튼을 누르면 1번 LED에만 불이 들어오게 된다.

SW2 : Stop watch를 reset 해주는 버튼. 버튼을 누르면 stop watch의 시간을 0으로 초기화 시킨다.

SW3 : Stop watch를 pause 해주는 버튼. 버튼을 누르면 stop watch의 시간을 count 하던 것을 멈추고 멈춘 시간을 출력한다.

SW4 : Stop watch를 start 해주는 버튼. 버튼을 누르면 stop watch의 시간을 count 하기 시작한다.

2) Text editor

FND : fpga의 switch가 현재 text를 입력하기 위해 몇 번 눌렸는지 count 한 값을 출력한다. 초기 상태는 0000. 9999를 넘어갈 경우, 만 단위는 생략한다.

SW : 알파벳과 숫자를 입력 받는 버튼

초기 상태는 알파벳 입력. 새로운 버튼의 입력이 한 번 들어올 때마다 한 글자씩 출력한다. 한번 눌렀던 버튼을 다시 누를 때마다 해당 알파벳을 입력 수에 맞게 바꾸어 준다. 즉 (2)번 버튼을 1번 누르면 A를 출력하고, (2)번 버튼을 3번 입력하면 C를 출력한다. (4), (5) 번 버튼의 입력이 한꺼번에 들어오지 않으면 text lcd에 출력 되는 string은 항상 덧붙여 출력한다.

(5), (6)번 버튼을 한꺼번에 누르면 알파벳 입력에서 숫자 입력으로 바꾼다. text lcd에 출력 되는 string의 값은 변하지 않고, 기존에 입력된 text에 덧붙여 입력한다. 한 번 누를 때마다 해당하는 숫자를 출력한다. 숫자 입력 시에는 같은 버튼을 여러 번 누르면 버튼을 누를 때마다 새로운 숫자를 출력한다.

버튼 2개를 동시에 누를 때, 각 기능을 수행한다.
(2), (3) : mode 변경, 이 버튼을 한꺼번에 누르면 mode 2에서 mode 3으로 변경된다.
(4), (5) : text lcd clear. 이 입력이 들어오면 text lcd에 있던 값을 없애고 다시 빈 상태로 만들어준다.
(5), (6) : 영어->숫자, 숫자->영어의 입력을 바꿔준다.
(8), (9) : 프로그램을 종료한다.

Text lcd : 스위치를 통해 생성된 text 값을 출력한다. 초기 상태는 lcd가 빈 상태.

Text lcd의 최대 출력 범위를 넘어가면 기존의 string에 가장 앞에 있던 문자를 제거하고 한칸씩 앞으로 밀고 나서, 새로 들어온 text를 출력한다. text lcd의 최대 출력 범위를 8이라고 가정하고 기존의 text를 ABCDEFGH라 할 때, 9번째 text인 W가 들어오면 BCDEFHW만 출력한다.

Dot matrix : 현재 switch로 받는 입력이 알파벳인지 숫자인지 나타낸다. 초기 상태는 A. 입력 받는 값이 알파벳이면 dot matrix에 A를, 입력 받는 값이 숫자면 dot matrix에 1을 출력한다.

3) 추가 구현

Text lcd : 사전에 지정된 텍스트를 왼쪽으로 shift 시키며 보여준다.

Dot matrix : 사전에 지정된 텍스트를 한글자씩 순차적으로 보여준다.

FND : SW 버튼을 통해 motor 와 buzzer를 제어한다.
SW1 : 모터를 active / inactive 상태로 변경한다.
SW2 : 모터의 진행 방향을 변경한다.
SW3 : Buzzer의 소리를 킨다.
SW4 : Buzzer의 소리를 끈다.

III. 추진 일정 및 개발 방법

자신들이 설정한 개발 목표를 달성하기 위한 개발 일정을 설정하고, 각 요소 문제를 해결하기 위해서 어떤 방법을 사용할 지 기술할 것. 또한 각 연구원의 역할을 분명히 기술할 것.

가. 추진 일정

‘14. 04. 08.

Shared memory를 이용해 input, main, output 프로세스 간의 기본적인 IPC 구조 구축
FND 드라이버를 이용해 mode 1의 stop watch 구현

‘14. 04. 10.

가독성을 위해 기존에 짠 코드 리팩토링
Stop watch mode 구현 완료

‘14. 04. 11.

Mode 2로 변경시 input process가 mode 2에게 붙잡혀 빠져나오지 못하는 문제 발견
Event key handler가 blocking input device 였음

‘14. 04. 12.

Input 프로세스를 2개로 나눔 (event key process, input process)
Modularity를 위해 코드를 re-factoring
Stop watch mode가 정확하게 작동
Text editor mode 작업 중
    Counting의 초기 값이 150으로 고정되어 있음
    입력이 조금 느리게 작동함
    LCD clear와 typing mode 작업 해야 함

‘14. 04. 13.

Counting이 정상적으로 작동
Text editor mode 정상적으로 작동
    여전히 반응 속도가 느리고, 누르고 있을 경우 count가 계속 올라감

‘14. 04. 15.

알고리즘 개선으로 typing 반응 속도 향상
할당하였던 shared memory 프로그램 종료 시 free 작업 수행
전체적인 성능 향상
Mode 3 작업 중

‘14. 04. 16.

Mode 3 완성 및 document 작성
Source code 정리 및 주석 미흡 부분 수정
Functions 최적화

나. 개발 방법

Process 간의 IPC 구현 : Shared memory 두 개를 이용하여 input process 에서 입력을 받아 shared memory를 통해 main process에게 전달하여 main process가 이를 계산하고 output process에 데이터를 전달해 출력하도록 구현

Input process : 각 드라이버를 열어 적절한 데이터를 입력 받아 shared memory를 통해 main process 에게 전달할 수 있도록 구현. 데이터를 넘겨 주고 main process가 연산을 끝냈다는 flag를 넘겨 받아야 다음 input을 받도록 해야 한다.

Main process : Input process로부터 입력 받은 데이터를 연산하여 적절한 결과 값을 output process에게 shared memory를 통해 전달하도록 구현. 연산이 완료되면 input process에게 연산이 끝났다는 flag를 전달한다.

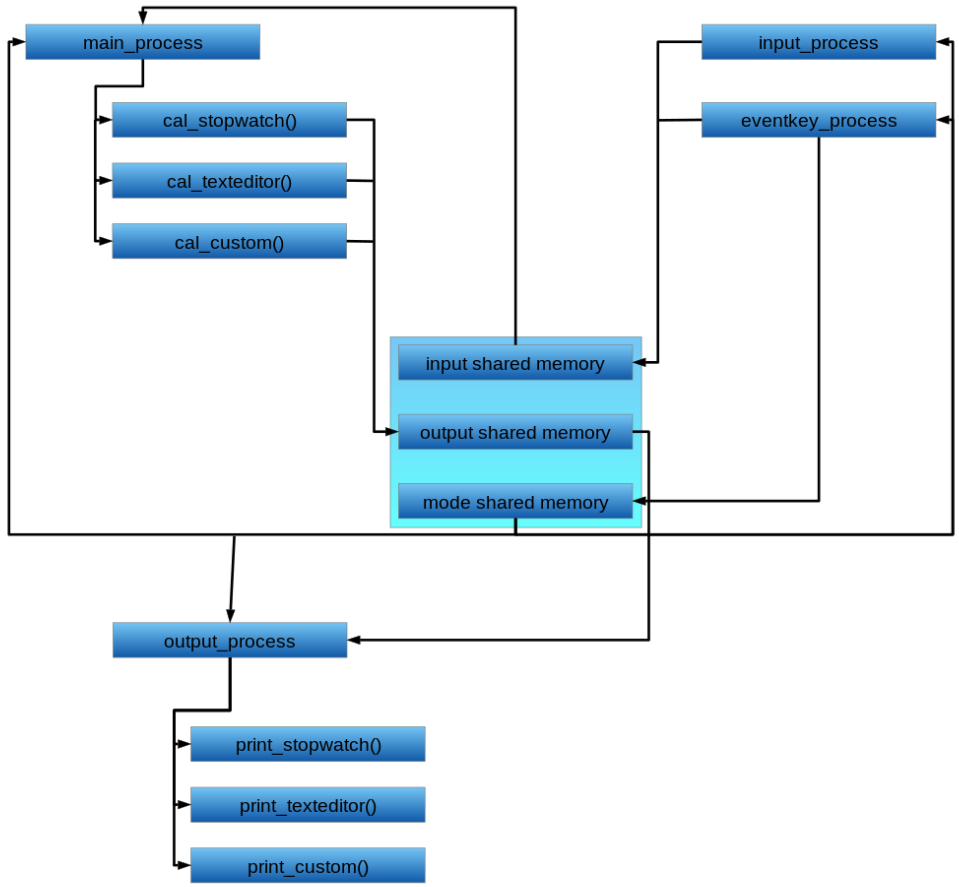
Output process : Main process로부터 전달 받은 데이터를 드라이버 장치에 출력해주도록 구현. 드라이버에 출력을 하고 난 후 main process에게 정상적으로 출력 하였다는 flag를 전달한다.

Mode 간의 전환 : Event key handler를 이용해서 어떠한 모드에 있더라도
즉각적으로 모드가 변경될 수 있도록 구현. 이때, 다른 작업이나 프로세스에 의해
지연되어서는 안됨

IV. 연구 결과

1. 합성 내용:

- 설계 목표에 필요한 내용을 조사 분석한 후 그들을 바탕으로 구성한 전체 프로그램 구성도.



2. 제작 내용: 개발 결과

static void die(char *str)

역할 : Driver를 open 하지 못했을 때 error 메세지를 출력하고 mode를 0으로 바꿔준다.

static void init_shared(void)

역할 : Input shared memory 와 output shared memory를 default 값으로 변경하여 준다.
구현 방법 : if 문을 통해 각 모드에 따른 default 값을 다르게 설정

static void free_shared(void)
역할 : 메모리에서 segment를 떼어내고 그 segment를 deallocate 시킨다.

static int shared_memory(void)
역할 : Mode, input, output에 해당하는 shared memory를 allocate하고 메모리에 붙인다.

static int main_process(void)
역할 : 각 모드에 따른 계산 함수를 호출한다.

int cal_stopwatch(void)
역할 : Input shared memory의 값을 통해 시간을 계산하고, output shared memory로 보낸다.
구현 방법 : Input shared memory에, SW3이나 SW4키가 들어오면 while 문을 지속적으로 돌려준다. 이때 difftime을 사용하여 1초와 근사한 값으로 while 문을 하나 더 내부에 돌려서 output shared memory에 데이터를 보내준다.

int cal_texteditor(void)
역할 : Push switch button으로 들어오는 값들을 분석해서 그에 맞는 값을 output shared memory로 전달하여 준다.
구현 방법 : 두 개의 키가 동시에 들어올 경우를 먼저 처리하여 주고, 아닐 경우에는 하나의 키로 인식을 하도록 구현한다.

int typing_mode(void)
역할 : 5, 6번 키가 동시에 눌렸을 경우 타이핑 모드를 변경해준다.
구현 방법 : Shared memory에 일부를 타이핑 모드로 할당하여, 상황에 맞는 값으로 변경해준다.

int typing_count(int count)
역할 : Push switch button이 몇 차례 눌렸는지 카운트 해준다.

int typing_clear(void)
역할 : 지금까지 적었던 string data를 초기화 시켜준다.

int typing_alphabet(void)
역할 : 알파벳 모드에서 타이핑 할 시 그에 맞는 character를 shared memory로 보내준다.
구현 방법 : 주어진 char 배열에서 빈 공간이 있는지 확인 한 후 이전에 입력 받은 글자와 그 글자의 index를 flag로 활용하여 동일한 버튼이 눌렸을 시 index를 증가 시켜 동일 버튼 내의 다른 글자로 바꾸어 주고 다른 버튼이 눌렸을 시 flag를 대체하고 새로운

글자를 string에 추가 시켜 준다. 이때, string buffer가 가득 찼을 시, for 문을 이용해 한 글자씩 왼쪽으로 이동 시켜 새로운 글자가 들어 올 수 있도록 하였다.

int typing_numeric(void)

역할 : 숫자 모드에서 타이핑 할 시 그에 맞는 숫자를 shared memory로 보내준다.
구현 방법 : switch 문을 이용해 해당하는 숫자의 ASCII 코드를 output shared memory에 입력해준다.

cal_custom(void)

역할 : 추가 구현 모드를 계산하고 결과 값을 output shared memory에 입력해준다.
구현 방법 : 미리 준비된 string을 한칸씩 왼쪽으로 shift 하여 문구가 이동하는 것처럼 보여준다. 또한 FND 드라이버의 SW1키와 SW2키로 모터의 움직임과 진행 방향을 제어할 수 있고, SW3과 SW4 키로 소리를 키고 끌 수 있다.

static int eventkey_process(void)

역할 : Event key로 들어오는 모든 input을 main process로 전달하여 준다.
구현 방법 : Input process와는 별개로 독립적인 process를 이용하여 항상 모드를 변경할 수 있도록 구현하고, 모드 1에 대한 input을 받을 수 있도록 한다.

static int input_process(void)

역할 : Evnet key를 제외한 모든 input을 main process로 전달하여 준다.
구현 방법 : Event key process와는 별개로 독립적인 process를 이용하여 모드2와 모드3에 대응하는 device driver로부터 데이터를 입력 받는다.

static int output_process(void)

역할 : 각각의 모드에 해당하는 print 함수를 호출한다.

int print_stopwatch(void)

역할 : Output shared memory에 있는 데이터를 FND driver에 출력해준다.

int print_texteditor(void)

역할 : Output shared memory에 있는 데이터를 fpga dot, fpga fnd, fpga text lcd 등에 출력해준다.

int print_custom(void)

역할 : Output shared memory에 있는 데이터를 fpga text lcd, fpga dot 에 출력해준다.

3. 시험 및 평가 내용:

요구 사항에 대해 모든 절차를 준수하였으며, 각각의 테스트 케이스들을 모두 통과하였다. Stop watch의 경우 약 6시간을 돌려 테스트 해 본 결과, 아무런 이상 없이 정상적으로 작동 되었고, 다른 기능에 전혀 영향을 주지 않았다. Text editor의 경우 초반에 typing에서 약간의 delay와 함께 드물게 버튼 인식이 되지 않는 경우가 있었는데 입력 알고리즘을 대폭 개편하고 난 후 매우 정상적으로

작동하고 있으며, 생각할 수 있는 거의 모든 케이스들을 직접 테스트 해 보았고, 문제를 발견할 수 없었다. 추가 구현 모드에서는 모드 1, 2에서 사용하지 않았던 드라이버를 간단하게 사용해 보는 정도로 마무리 지었기 때문에 따로 문제점을 발견할 수 없었다.

처음부터 기능 구현에 집중하지 않고, 각각의 프로세스와 shared memory 간의 통신에 집중을 해서 여러 차례 테스트 끝에 기본 구조 틀을 완성하고 드라이버를 function 별로 관리를 하였기 때문에 추후에 문제가 생긴다 하더라도 짧은 시간 내에 문제점을 찾고 수정할 수 있으며, 각각의 function 간에는 static을 사용하여 접근을 제한하였기 때문에 안정성 및 내구성 등에 대해서는 만족스러웠다. 모듈화를 통해 추가적인 기능도 큰 문제 없이 추가할 수 있도록 하여, 추후에 새로운 기능 추가 시 더 좋은 performance를 보여줄 수 있을 것이라 기대된다.

V. 기타

1. 연구조원 기여도:

20091648 이준호 : 100%

2. 기타 본 설계 프로젝트를 수행하면서 느낀 점을 요약하여 기술하라. 내용은 어떤 것이든 상관이 없으며, 본 프로젝트에 대한 문제점 제시 및 제안을 포함하여 자유롭게 기술할 것.

기존에 사용하고 있던 스마트폰 및 smartpad의 application과 다르게 기기 내의 드라이버에 직접 접근하여 드라이버를 컨트롤하고 IPC를 통해 프로세스 간에 메모리를 공유하는 부분이 신기하고 재미있었다. 제한된 성능에서 구현을 하는 것이기 때문에 코드를 짤 때에도 좀 더 신중하고 세세하게 설계하여 구현을 하게 되었는데, 그 결과가 만족스러워서 더 괜찮았던 것 같다. 다만 추가 구현 모드에 대해서 명확하게 어느 정도 이상을 해야 한다는 명시가 없어서 구현 상에 약간의 어려움이 있었다.