ARM 프로세서를 이용한 직렬과 병렬 데이터 통신

최 원 호*, 황 욱 철*, 정 민 수*

*경남대학교 컴퓨터공학과

A Serial and Parallel Data Communication Using ARM Processor

Won-Ho Choi*, Wook-Chul Hwang*, Min-Soo Jung*

*Dept. of Computer Engineering, Kyungnam University

요 약

ARM 프로세서는 CISC 보다는 간단하게 디자인된 RISC 로서 내장 응용프로그램에 적합하기 때문에 앞으로 모든 디지털 기기에 ARM 코어를 기반으로 한 핵심 칩들이 생산된다. 그러나 명령어가 CISC 보다는 적기 때문에 주어진 작업에 대해 완전한 처리를 위해서는 보다 많은 명령어들을 필요로 한다. 이러한 ARM 프로세서에서 데이터를 전송할 때 사용하는 메모리 영역과 레지스터들을 프로그램과 함께 분석하였다.

1. 서 론

최근 통신 수단으로는 셀룰러폰과 개인휴대통신 등을 가장 많이 이용한다. 대부분의 이동단말기는 표준 내장 칩인 ARM(Advanced RISC Machine) 프로세서와 CDMA를 이용한다. ARM 프로세서는 고성능 저전력이라는 특징을 가지고 있기 때문에 이동단말기뿐만 아니라 MSM3000 과 ITM2000 에서도 표준 프로세서로서 이용되고 있다. ARM 프로세서는 보다 향상된 RISC 의 기능들을 가지고 있으며, 적재-저장 구조, 고정된 32 비트 명령과 3-주소 명령 형식을 사용한다.

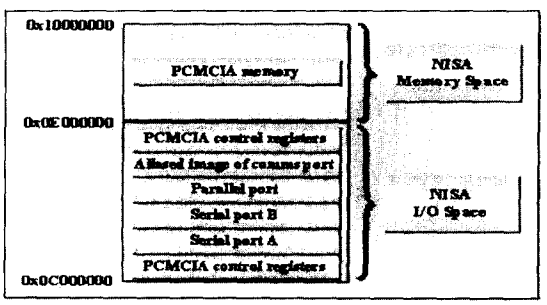
본 논문은 ARM 프로세서가 직렬 포트와 병렬 포트를 이용해서 호스트 시스템과 데이터를 교환하는 방법에 대해서 분석하였다. 본 논문의 구성은 2 장에서 ARM 프로세서가 호스트 시스템과 통신을 위해서 사용하는 메모리에 대해서 언급하였고, 3 장에서 직렬 포트를 이용할 때에 데이터를 전송하는 방법을 설명하였다. 4 장에서 병렬 포트를 사용하여 필요한 데이터를 전송하는 방법을 분석하였다. 마지막으로 본 논문에 대한 연구 결과와 향후 연구 방향에 대해서 기술하였다.

2. ARM 프로세서가 사용하는 메모리

ARM 프로세서에서 통신 포트에 대한 기본 주소들은 NISA(Not ISA) 인터페이스에서 암호화되고, 데이터를 전송하기 위해서 단지 하위 8 비트만을 사용하지만 통신 포트의 레지스터들은 워드 형태이다. NISA 메모리의 구성은 그림 1 과 같다.[1,2]

직렬과 병렬 입출력 시스템은 ST16C552 장치에 기반을 두고 있다. 이 입출력 시스템은 16Byte 송신과 수신을 갖는 이중 비동기 수신자와 송신자이고, 양방향 센트로닉스(Centronics) 형태의 병렬 프린트 포트이다.[3,4]

직렬포트 레지스터가 사용하는 주소값은 표 1 과 같다.



[그림 1] NISA 메모리 영역

[표 1] 직렬 포트 레지스터에 대한 주소

직렬포트 A	직렬포트 B	Read 모드	Write 모드
0x0D800000	0x0D800020	Rx Holding	Tx Holding
0x0D800004	0x0D800024		Interrupt Enable
0x0D800008	0x0D800028	InterruptStatus	FIFO Control
0x0D80000C	0x0D80002C	Line Control	
0x0D800010	0x0D800030		Modem Control
0x0D800014	0x0D800034	Line Status	
0x0D800018	0x0D800038	ModemStatus	
0x0D80001C	0x0D80003C	Scratchpad	Scratchpad
0x0D800000	0x0D800020		DivisorLatchLSB
0x0D800004	0x0D800024		DivisorLatchMSB

병렬 포트 레지스터가 사용하는 주소 값은 표 2 와 같

다.

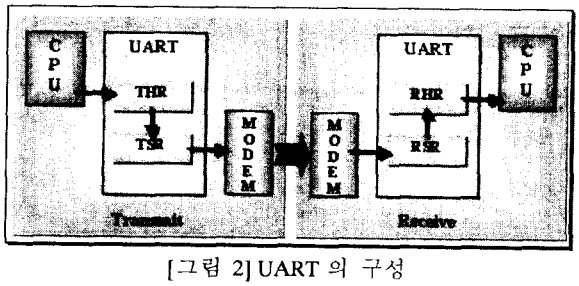
[표 2] 병렬 포트 레지스터에 대한 주소

주소	Read 모드	Write 모드
0x0D800040	Data	Data
0x0D800044	Status	I/O Select
0x0D800048	Command	Control

3. 직렬 포트를 이용한 데이터 통신

직렬 포트 통신에 사용되는 ST16C552 장치는 16 바이트 송신과 수신 큐(Queue)를 가지고, UART(Universal Asynchronous Receiver / Transmitter)를 통해 데이터 통신을 한다. 그리고, ST16C552 는 수행되고 있는 에러 조건, 타입, 전송 동작의 상태 등을 상태 레지스터에 제공한다[5].

UART 는 시작, 정지 패리티 비트들을 생성하고 제거하는 것을 책임진다. 정지 조건, 프레임 구성, 범람 오류 등의 채널 상태가 UART 에 의해 프로세서로 전달되고, 프로세서는 회선 속도, 단어 크기, 패리티, 정지 비트의 개수 등의 제어신호를 UART 로 보낼 수 있다. 데이터를 전송할 때, UART 는 병렬 직렬 변환을 사용하여 내부의 병렬 바이트를 직력의 비트 흐름으로 변환해야 한다. 전송될 바이트들은 프로세서에 의해 UART 로 공급된다. 그 바이트들은 자리 이동 레지스터(Shift Register)에 저장된다. 시작 비트를 먼저 보내고, 자리 이동 레지스터에 있는 비트들을 뒤따라 보낸다. 자리 이동 레지스터는 회선 속도에 따라 한번에 하나씩 채널로 비트들을 보낸다[4].



[그림 2] UART 의 구성

데이터 전송은 프로세서가 각 레지스터들의 값을 보고 데이터 비트들을 UART 로 보낼 것인지를 결정되고, 각 레지스터들은 하위 8 비트의 값에 의해 수행 처리가 된다[2,5].

[표 3] 직렬 포트가 사용하는 레지스터

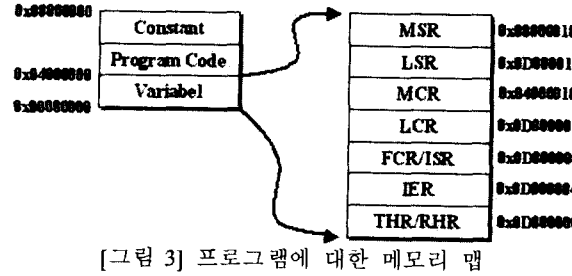
직렬포트 A	직렬포트 B	READ 모드	WRITE 모드
0x0D800000	0x0D800020	RHR	THR
0x0D800004	0x0D800024		IER
0x0D800008	0x0D800028	ISR	FCR
0x0D80000C	0x0D80002C		LCR
0x0D800010	0x0D800030		MCR
0x0D800014	0x0D800034	LSR	
0x0D800018	0x0D800038	MSR	
0x0D80001C	0x0D80003C	SPR	SPR
0x0D800000	0x0D800020		DLL
0x0D800004	0x0D800024		DLM

3.1 직렬 포트 레지스터들

- **Interrupt Eanble Register(IER)**
  데이터 수신 준비, 송신 데이터 부재, 라인 상태, 모뎀 상태 레지스터로부터 발생할 수 있는 인터럽트들을 허용 하는데 사용하는 레지스터이다.
- **Interrupt Status Register(ISR)**
  인터럽트 처리에 우선 순위를 결정하는 사용되는 레지스터이다.
- **FIFO Control Register(FCR)**
  이 레지스터는 FIFO 를 허용하고, FIFO 를 지우고, 수신 FIFO 트리거 단계를 설정하고, DMA 신호의 형태를 선택하기 위해서 사용된다.
- **Line Control Register(LCR)**
  LCR 은 비동기식 데이터 통신 형식을 명세하기 위해서 사용된다. 워드 길이의 수, 스톱 비트, 패리티 등은 레지스터 내에서 임의의 비트를 설정함으로써 선택되어진다.
- **Modem Control Register(MCR)**
  모뎀이나 주변기기를 갖는 인터페이스를 제어하기 위해서 사용되는 레지스터이다. 이 레지스터에서 비트-4 는 정상적인 모드로 동작할 것인지, 루프-백(loop-back) 모드로 동작할 것인지를 결정하는데 사용된다.
- **Line Status Register(LSR)**
  이 레지스터는 CPU 에 데이터 전송의 상태를 제공한다. 비트-0 은 전송할 데이터의 유무를 나타내는 데이터의 대기 상태를 표시하고, 비트-2 는 오버런(overrun) 에러가 발생했는지를 상태를 나타낸다.
- **Modem Status Register(MSR)**
  모뎀이나 주변기기로부터 CPU 에 제어 라인들의 현재 상태를 제공하는데, 내부에 변화된 상태들이 있으면 각 해당 비트들은 '1'로 값이 설정되고, 변화된 상태를 CPU 가 읽어가면 비트들은 다시 '0'으로 설정된다.

3.2 직렬 포트를 이용한 프로그램

아래의 프로그램은 직렬 포트 A 와 B 간의 데이터 통신을 할 수 있도록 만든 프로그램이다. 이 프로그램에서는 양쪽 직렬 포트가 수행하는 일들이 같기 때문에 한쪽 포트에 대해서만 기술하였다.



[그림 3] 프로그램에 대한 메모리 맵

- **송신 루프와 수신 루프**
  모든 전송 동작이 완전히 끝날 때까지 루프가 수행이

되고, 성공적으로 통신이 수행되기 위해 모든 포트들을 초기화한다.

각 포인터 변수들은 직렬 포트 레지스터들을 참조하고 있다. 그리고 송신 메소드는 먼저 전송할 문자열이 마지막 문자인지를 체크하고, THR 의 상태를 나타내는 LSR 의 값에 따라 문자열을 THR 로 이동시킨다. 문자열의 이동이 끝나면 THR 에서 RHR 로 데이터를 전송한다. 수신 메소드는 전송된 문자를 RHR 에 저장시키고, 이를 프로세서에게 알린다.

```
int Serial(void)
{ *SerA_FCR = FCR_Fifo_Enable | FCR_Rx_Fifo_Reset |
              FCR_Tx_Fifo_Reset ;
  *SerA_LCR = LCR_Divisor_Latch ;
  *SerA_DLL = DLL_50_Baud;
  *SerA_DLM = DLM_50_Baud;
  *SerA_LCR = LCR_8_Bit_Word_1 ;
  *SerA_MCR = MCR_Loopback_Mode;

  while ( Serial_A_Tx == 0) {
   if (!TxFifoFull(SerA_LSR))
     { Serial_A_Tx= TEST_PATTERN ; *SerA_THR =Serial_A_Tx ; }
  }

  while (Serial_A_Rx == 0) {
   if (!RxFifoEmpty(SerA_LSR)) { Serial_A_Rx = *SerA_RHR; }
  }
```

4. 병렬 포트를 이용한 데이터 통신

직렬 포트가 UART 를 사용해서 데이터 전송을 하는 반면에 병렬 포트는 양방향 센트로닉스 형태의 프린터 포트를 이용해서 데이터 통신을 한다. 데이터 전송을 위해 사용하는 병렬 포트 레지스터들은 표 4 와 같고 병렬 포트는 레지스터의 비트 값들을 이용해서 데이터가 입력 값인지 출력 값인지를 결정한다[6].

[표 4] 병렬 포트의 데이터 전송에 사용되는 레지스터

주소	Read 모드	Write 모드
0x0D800040	Port Register	Port Register
0x0D800044	Status Register	I/O Select Register
0x0D800048	Command Register	Control Register

4.2 병렬 포트 레지스터들

- **Port Register**

  포트 레지스터는 송· 수신할 데이터를 저장하는 레지스터이다.

- **Status Register**

  상태 레지스터는 출력 상태와 인터럽트 상태를 프로세서에 제공한다.

- **Command Register**

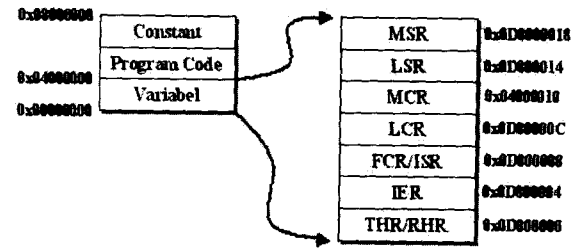
  명령 레지스터는 입· 출력 방향에 관계없이 스토브, 자동입력, 초기화, 선택입력 그리고 인터럽트 허용 비트 등의 입력 상태를 읽어 들인다.

- **Control Register**

  제어 레지스터는 비트-5 의 값의 설정에 따라 병렬 포트가 입력을 수행할 것이지 출력을 수행할 것이지를 결정한다.

4.3 병렬 포트를 이용한 프로그램

이 프로그램에서 사용하는 레지스터에 대한 메모리 맵은 다음과 같다.



[그림 4] 프로그램에 대한 메모리 맵

병렬 포트 레지스터를 이용한 데이터 통신은 포트 레지스터에 있는 데이터를 제어 레지스트의 비트 값에 의해 송신과 수신의 방향이 결정된다. 그리고 송수신을 구별하기 위해서 특정 시간동안 루프를 수행한다.

```
int Parallel(void)
{
 *ParPR = 0; */
 while (INFINITE_LOOP) {
  Command_Value = *ParCom;
  *ParCon = Command_Value | PCNT_IO_Select;

  for (Delay_Count=0;Delay_Count<=WRITE_SETUP_DELAY;
       Delay_Count++){ }

  Switch_Value = *ParPR & SWITCH_NIBBLE_MASK;
  Command_Value = *ParCom;
  *ParCon = Command_Value & ~(PCNT_IO_Select);
 }
   return 0;
}
```

5. 결론 및 향후 연구 과제

ARM 프로세서를 이용해서 호스트 시스템과 직렬 및 병렬 포트를 이용한 데이터 전송에 있어 필요한 메모리 영역과 각각의 레지스터의 수행 과정을 분석할 수 있었다. 그리고 위의 두가지 프로그램 구현을 통해 데이터 통신을 위한 레지스터를 제어할 수 있었다. 본 연구를 바탕으로 향후 연구 과제로는 액정디스플레이(LCD) 장치에 ARM 프로세서를 이용해서 필요한 정보들을 출력시킬 수 있게 제어할 수 있는 방안에 대해서 연구를 할 것이다.

참고 문헌

[1] http://www.arm.com
[2] Steve Furver, "ARM System Architecture", Addison-Wesley
[3] 윤현수, "데이터통신", 정익사
[4] 짐종상 역, "데이터통신 및 컴퓨터통신", 선중당
[5] ARM Limited, "ARM datasheets"
[6] van Someren and Atack, "The ARM RISC Chip"
[7] AMBA Specification(ARM IHI 0001)
[8] Reference Peripherals Specification(ARM DDI 0062D)