의공학회지 : 제18권, 제4호, 1997
J. of KOSOMBE : Vol. 18, No. 4

연 구 논 문
97-18-4-09

시각 프로그래밍이 가능한 신호분석 환경

박승훈·우응제·이헌주*·황진하**·김형진***·장재명****

= Abstract =

Signal Analysis and Visualization Environment with Visual Programming Capability

Seung Hun Park, Eung Je Woo, Hun Joo Lee*, Jin Hah Hwang**,
Hyung Jin Kim***, Jae Myoung Jang****

In this paper, we present a signal analysis environment with visual programming capability, which is called Signal Analysis and Visualization Environment(SAVE). The system allows a user to perform visually programmed analysis of signals and visualize the results. The visualization facility enables the user to compare original signals with processed ones and also to display signals overlaid synchronously with the events extracted from the signals. The SAVE system has an extensible architecture: each signal processing algorithm is implemented as a separate building block object module, which can be freely added or removed from the SAVE system without any code modification. We describe the overall structure of the SAVE system and the building block objects, which provide the extensibility in collaboration with together. We illustrate some test runs in order to give a taste of how to use and where to use the system.

Key words : Signal visualization, Signal analysis, Signal processing

서 론

대부분의 신호 수집 및 처리 시스템은 특정 응용 분야에 한정된 신호 처리 기법을 사용하여 실시간으로 신호를 처리하거나, 저장 매체나 종이에 기록한 다음 육안이나 컴퓨터를 사용하여 처리한다. 의료, 산업, 교육 등 각 분야에서 사용되는 이러한 신호 분석 시스템들은 특정 응용 분야에 적합하도록 개발되어, 사용자의 요구가 달라졌을 때, 이에 적합하도록 수정하기가 힘들다. 또한, 그 시스템을 사용하기 위해서는 미리 많은 준비와 교육이 필요하다. 신호 처리를 담당하는 개발자들은 시스템을 설계할 때 문제점을 발견하기 위해 시간이 많이 드는 구현 과정을 거친 후에야 비로소 설계의 문제점을 발견할 수 있다.

특정한 신호 성분을 검출하는 알고리즘을 개발하거나, 어떤 알고리즘에서 적합한 계수들의 값을 찾아야 하는 경우, 시행 착오에 의한 방법을 많이 사용한다. 즉, 계수들을 바꾸어 결과를 본 후, 가장 좋은 계수의 조합을 선택한다. 이러한 시행 착오 방식은 많은 시간과 노력을 필요

*건국대학교 의과대학 의공학과
Department of Biomedical Engineering, College of Medicine, Kon Kuk University
* 나눔기술
* Nanum Technology
** 싱크론 기술
** Synchron Technology
*** 삼성종합기술원 의료기기팀
*** Samsung Advanced Institute of Technology, Medical Electronics Team
**** CU 정보기술
**** CU Information Technolgy
본 연구는 1996년도 학술진흥재단 지방대학 육성 연구사업의 지원에 의한 결과임.(과제번호 : 02-F-0215)
통신저자 : 우응제, (380-701) 충북 충주시 단월동 건국대학교 의과대학 의공학과, Tel. (0441)840-3762, Fax. (0441)851-0620

로 한다. 따라서, 이러한 작업을 손쉽게 하기 위한 소프트웨어적인 작업 환경에 대한 연구가 수행되어 왔다. 생체 신호의 관찰 및 알고리즘의 최적 계수들의 집합을 육안 관찰에 의해 찾을 수 있는 작업 환경에 대한 연구와 여러 채널의 뇌파를 관찰하고, 저장하며, 나중에 검색할 수 있는 시스템이 개발되었다[1-4]. 이러한 시스템들은 고정된 메뉴나 정해진 철자법에 의해 원하는 신호 처리 알고리즘을 표현하며, 변경이나 확장이 어렵다. 따라서, 사용자는 프로그래밍을 하지 않는 대신 신호 처리 알고리즘 표현 언어를 별도로 배워야 한다.

최근 연구가 활성화 되고 있는 시각 프로그래밍 방법은 사용자가 프로그래밍 언어나 컴퓨터에 대한 지식 없이도 자신이 원하는 것을 손쉽게 나타낼 수 있는 융통성 있는 작업 환경을 제공한다. 본 연구에서는 신호 처리 기능 블록들을 시각 프로그래밍 기법으로 조합하여 원하는 신호 분석 작업을 수행할 수 있는 신호 분석 환경 시스템을 개발하였다. 기본적인 신호 처리 알고리즘들을 기능 블록의 형태로 내장하였고, 사용자가 개발한 신호 처리 알고리즘들을 정해진 형식에 맞추어 기능 블록의 형태로 구현하기만 하면, 코드의 변경 없이 신호 분석 환경에 추가할 수 있다. 기본적으로 제공되는 시각화 기능 블록들을 사용하면, 원래의 신호들과 신호 처리 알고리즘에 의해 변화된 신호들을 비교, 관찰할 수 있을 뿐 아니라, 신호 검출 알고리즘에 의해 검출된 사건 정보들을 신호에 중첩하여 관찰할 수 있도록 하였다.

신호 분석 환경의 기본 요건과 접근 방법

프로그래밍 비전문가들이 짧은 시간 동안에 여러 방법으로 신호들을 분석하고, 서로 비교하여 최적의 알고리즘과 계수들을 선택할 수 있는 신호 분석 환경을 제공하기 위해 본 연구에서 설정한 신호 분석 환경 시스템이 갖추어야 할 기본적인 요건들은 다음과 같다.

- 간단한 마우스의 조작으로 프로그램의 재구성이나 기능의 변경이 가능해야 한다.
- 입출력이 연결되지 않았거나 연결될 수 없는 곳에 연결되는 경우와 같은 알고리즘 구성상의 문제점을 자동으로 검출하여 알려 주어야 한다.
- 신호의 시각화 기능이 다양하게 제공되어야 하고, 새로 추가할 수 있어야 한다.
- 시간 정보나 발생한 사건에 대한 정보를 사용하여 특정 부분을 검색할 수 있어야 한다.
- 신호 처리 알고리즘의 계수들을 실행하는 도중에 변경할 수 있어야 한다.
- 다양한 형식의 파일을 읽을 수 있어야 하며, 새로운 형식의 파일이 나타났을 때 쉽게 수용할 수 있어야 한다.
- 알고리즘을 설계하는 환경에서 신호 처리 결과를 실제로 관찰할 수 있는 실행 환경으로, 그리고 그 역으로의 전환이 쉬워야 한다.

신호 분석 환경에서는 신호 분석을 위한 알고리즘의 설계와 그 결과를 바로 관찰할 수 있는 실행 환경이 필요하다. 프로그래밍 비전문가가 신호 처리 알고리즘을 빠른 시간 안에 재구성하여 그 결과를 확인하기 위해서는 프로그래밍 언어에 의한 알고리즘 구현 방법은 적당하지 않다. 비전문가들이 직관적인 방법으로 프로그래밍할 수 있는 방법들 중에서 쉽게 접근할 수 있는 방법은 시각 프로그래밍 방법이다. 시각 프로그래밍 방법은 정해진 철자법을 갖는 언어를 사용하지 않고 사용자가 시각적인 직관으로 자신의 의도를 쉽게 표현할 수 있는 프로그래밍 방법으로, 추상화가 많이 이루어진 기본 요소들을 간단한 흐름 제어를 사용하여 원하는 작업을 수행하도록 조합한다. 본 연구에서는 시각 프로그래밍 기능을 제공하여 신호 처리 작업을 정의할 수 있는 신호 분석 환경 시스템을 개발하였다. 본 연구에서 수립한 설계 목표는 다음과 같다.

- 신호 처리 요소들, 외부 및 파일 입출력 요소들, 시각화 요소들을 기능 블록 객체라 불리우는 C++ 객체로 구현한다. 기능 블록 객체의 설계와 실행 과정에 사용되는 인터페이스들은 메쏘드(method)의 형태로 구현한다. 각 기능 블록 객체는 실행시점에서 동적으로 연결하여 실행할 수 있는 독립적인 Dynamic Link Library(DLL) 모듈 형태로 구현하여 확장과 통합을 쉽게 한다.
- 기능 블록 객체를 연결하는 선은 C++ 객체인 연결 객체의 형태로 구현하고, 자동으로 연결선을 배치할 수 있는 메쏘드를 갖게 한다.
- 기능 블록의 입력과 출력 부분은 C++ 객체인 입출력 포트 객체로 나타내고, 기능 블록의 왼쪽과 오른쪽에 나타나게 한다.
- 각 기능 블록 객체는 계수들을 변경시킬 수 있는 고유의 대화상자를 갖도록 하고, 대화상자를 호출할 수 있는 메쏘드를 설계 인터페이스에 포함하게 한다.
- 신호 버퍼의 첫 번째 데이터 수집 시간, 표본화 주파수, 그리고 버퍼의 인덱스를 사용하여 시간 정보를 추출한다.
- 사건 정보는 사건이 발생한 시작 시간과 끝나는 시간을 포함하여 신호 데이터와 동기화가 쉽게 이루어 지도록 한다.
- 입출력 포트 객체는 신호 데이터와 해당 채널의 사건 정보들을 전달받거나 전달할 수 있도록 2종류의 버퍼를 설치한다.

- 설계 환경에서 기능 블록 객체를 삽입할 때 실제로 메모리에 존재하는 객체로 생성하여 실행 환경에서 바로 사용할 수 있게 한다.
- 기능 블록 객체들에게 입력의 연결성을 검사하는 메쏘드를 설계 인터페이스에 포함하여 입력 연결 오류를 각 객체가 스스로 검출할 수 있게 한다.
- 기능 블록 객체 DLL들은 특정 디렉토리에 저장하게 하여, 현재 사용 가능한 기능 블록 객체들을 쉽게 파악할 수 있게 한다.
- 시각화 기능 블록 객체를 취향에 맞추어 작성한 다음, 추가할 수 있도록 한다. 시간 영역에서 신호를 시각화 할 수 있는 기능 블록 객체는 신호를 화면에 출력할 수 있을 뿐 아니라, 신호에서 추출한 사건 정보를 신호와 중첩하여 출력할 수 있는 기능을 구비한다. 신호의 특정 부분을 확대하는 기능, 신호의 chart speed, sensitivity 등을 조절할 수 있는 기능, 신호의 특정 부분을 블록으로 지정하여 그 부분의 최대, 최소값과 시간 정보, peak-to-peak의 진폭 및 시간 간격 등을 측정하는 기능을 갖추도록 한다.
- 신호 데이터를 시간을 이용하여 검색할 수 있으며, 사건 검색 기능을 이용하여 특정 사건이 발생한 지점으로 이동할 수 있도록 한다.

신호 분석 환경 시스템의 설계

객체 지향 개념에 의거하여 설계된 시스템은 그 시스템의 구성 요소인 객체들의 속성과 동작을 나타내는 클래스에 대한 설명과 상속 관계, 그리고 객체들의 종류와 상호 관계들을 기술함으로써 묘사될 수 있다. 본 논문에서는 신호 분석 환경 시스템에 존재하는 중요한 객체들의 클래스들과 그들의 상속 관계에 대해 기술한 다음, 실행 중에 존재하는 객체들 간의 상호 작용들을 기술함으로써 시스템의 구조와 동작에 대해 설명하였다.

1. 주요 클래스들의 종류와 상호 관계

신호 분석 환경 시스템은 신호 분석 작업을 설계하는 작업 환경 모듈, 설계된 신호 분석 작업을 실제로 실행하여 그 결과를 검증할 수 있는 실행 환경 모듈, 그리고 신호 분석 작업의 기본 구성 요소인 기능 블록 모듈들로 이루어진다.

설계 환경 모듈을 구현하기 위해 필요한 클래스들은 전체 시스템의 제어를 맡고 있는 SaveMainFrame 클래스, 현재 사용할 수 있는 기능 블록 객체들에 대한 정보를 관리하는 BlockObjectManager 클래스, 마우스를 사용하여

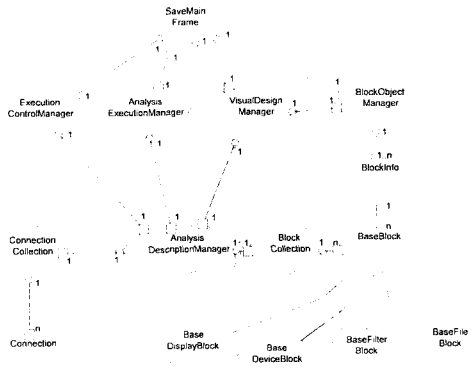

그림 1. 주요 클래스들의 종류와 상호 관계
Fig. 1. Relationship of major classes

기능 블록 객체들의 아이콘들을 조합하여 수행하고자 하는 신호 분석 작업을 설계하는 VisualDesignManager 클래스이다. 실행 환경 모듈에는, 신호 분석 작업의 실행을 담당하는 AnalysisExecutionManager 클래스, 신호 입력을 제어하는 InputControlManager 클래스가 존재한다. 설계 환경 모듈과 실행 환경 모듈이 공통으로 사용하는 클래스로는 AnalysisDescriptionManager가 있다. AnalysisDescriptionManager 클래스는 BaseBlock 클래스 객체들과 그것들을 연결하는 Connection 클래스 객체들을 모아서 관리하는 BlockCollection클래스와 ConnectionCollection 클래스를 포함하고 있다.

기능 블록 객체들을 구현하기 위해서 BaseBlock 클래스, BasePort 클래스, 여기에서 상속된 InputPort 클래스와 OutputPort 클래스 등이 필요하다. BaseFileBlock 클래스나 BaseDeviceBlock 클래스, 그리고 BaseFilterBlock 클래스 등은 BaseBlock 클래스로부터 기능 블록의 공통적인 속성과 시각 프로그래밍 인터페이스를 상속 받는다. 그림 1은 이러한 클래스들의 상호 관계를 Booch 방법을 사용하여 나타낸 것이다[5].

2. 주요 객체들의 상호 작용

객체 지향 프로그래밍 개념으로 구현된 시스템의 동작은 객체들의 상태 변화와 객체들간의 메시지 교환으로 나타낼 수가 있다. 그림 2는 주요 객체들의 초기 생성 과정을 나타낸 것이다. 신호 분석 환경 시스템의 중심 객체인 SaveMainFrame클래스의 인스탄스(instance)인 SaveMain 객체는 시동 단계에서 주요 객체들을 생성하고, 서로 연결시켜주는 역할을 담당하고 있다. 우선, 기능 블록들의 정보와 생성을 담당하는 BlockObjectMan 객체를 생성하고, AnalysisDescriptionMan, VisualDesignMan,

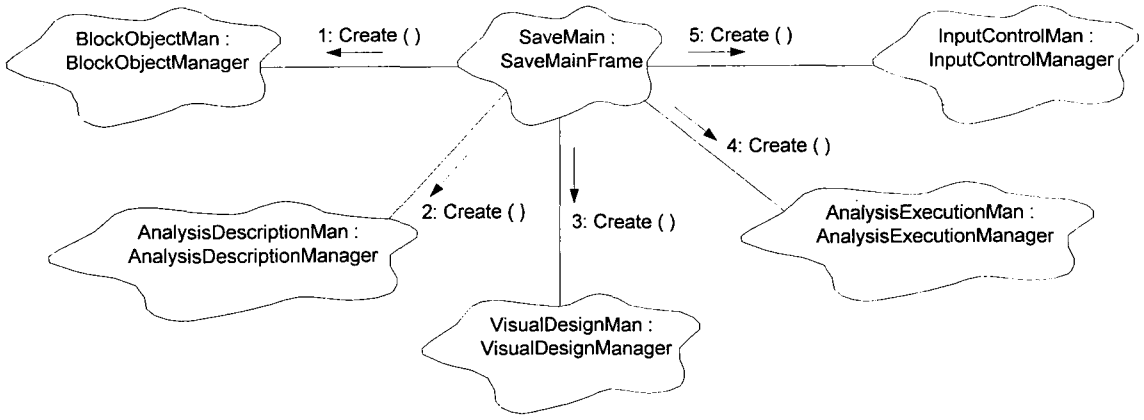

그림 2. 시스템의 초기 생성 과정
Fig. 2. Object instantiation in the system initialization stage

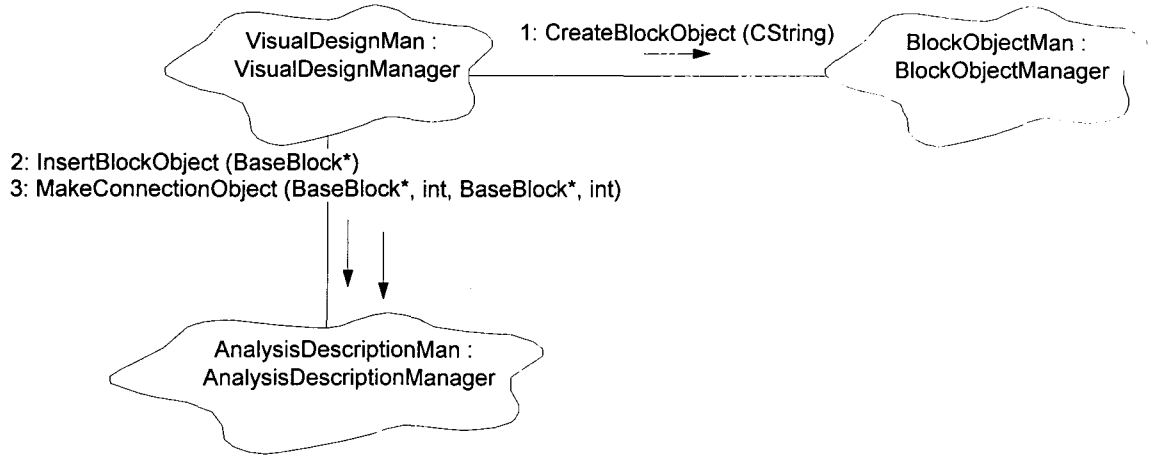

그림 3. 신호 분석 작업을 설계하는 경우의 메시지 교환
Fig. 3. Exchange of messages in the design of signal analysis job

AnalysisExecutionMan, InputControlMan 객체들을 차례로 생성한다. 각 객체들에게 관련 있는 객체들의 접속점에 대한 정보를 제공하여, 직접 접촉할 수 있도록 한다.

실제 환경에서 신호 분석 작업을 설계하는 과정은 사용자가 GUI(Graphic User Interface)를 사용하여 원하는 기능 블록을 나타내는 아이콘들을 도구상자에서 한 개씩 선택하여 설계 작업 공간에 끌어서 옮겨 놓은 다음, 소스(source)와 싱크(sink)의 관계를 나타내는 연결선을 사용하여 그들을 서로 연결하는 작업으로 이루어져 있다.

신호 분석 작업의 설계 과정에서 이루어지는 BlockObjectMan객체, AnalysisDescriptionMan 객체 및 Visual-DesignMan 객체들 사이의 메시지 교환은 그림 3에 간략하게 나타나 있다.

BlockObjectMan객체는 생성될 때, 특정 디렉토리를 검색하여 사용할 수 있는 기능 블록에 대한 정보를 Visual-DesignMan객체에 제공하여, 그것으로 하여금 사용할 수 있는 기능 블록 도구 상자를 생성할 수 있게 한다. 사용자가 도구 상자에 있는 특정 기능 블록 아이콘을 선택하여 설계 작업 공간에 끌어 옮겨 놓으면, VisualDesignMan 객체는 BlockObjectMan객체에 그 기능 블록의 이름과 함께 CreateBlockObject 메시지를 주어 새로운 기능 블록 객체를 생성하게 한 다음, AnalysisDescriptionMan객체에 그 객체에 대한 포인터와 함께 InsertBlockObject 메시지를 보내 삽입한다. VisualDesignMan객체는 삽입된 기능 블록 객체들의 데이터 소스와 데이터 싱크 관계를 나타내기 위해서 관련된 2개의 기능 블록 객체들에 대한 포인터와 정수로 표현되는 입출력 포트의 번호를 MakeConnectionObject 메시지와 함께 AnalysisDescriptionMan 객체에 보낸다. 이러한 작업을 되풀이하여 원하는 신호 분석 작업을 정의할 수 있다. 정의된 신호 분석 작업은 AnalysisDescriptionMan객체에 포인터를 사용한 연결 네트워크의 형태로 저장된다.

설계된 신호 분석 작업을 실행하는 과정에서 이루어지는 AnalysisExecutionMan객체, AnalysisDescriptionMan

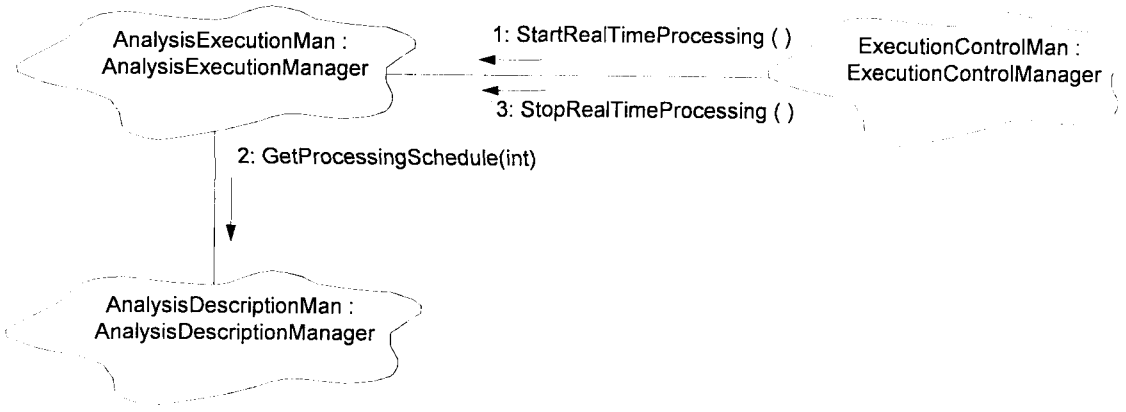

그림 4. 실시간으로 신호를 수집하고 분석하는 경우의 메시지 교환
Fig. 4. Message exchange of messages in the real-time signal acquisition and analysis

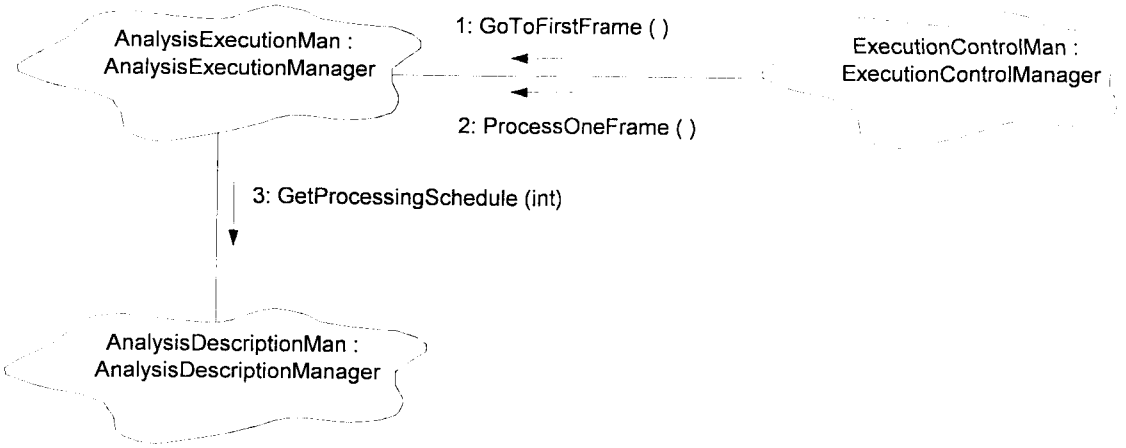

그림 5. 저장된 신호를 분석할 경우의 메시지 교환
Fig. 5. Message exchange in the analysis of stored signal data

객체 및 ExecutionControlMan 객체들 사이의 메시지 교환은 그림 4와 그림 5에 간략하게 나타나 있다. 그림 4는 실시간으로 신호를 수집하여 분석할 때의 메시지의 교환을 나타낸 것이고, 그림 5는 이미 파일에 저장된 신호를 분석할 때의 메시지의 교환을 나타낸 것이다.

ExecutionControlMan 객체는 GUI를 통해 사용자의 제어 입력을 받아 신호 분석 작업의 실행을 제어한다. 실시간으로 신호를 수집하면서 분석하고 그 결과를 화면에 출력하는 경우에는 AnalysisExecutionMan 객체에 StartRealTimeProcessing이라는 메시지를 보내고, 이를 받은 AnalysisExecutionMan 객체는 AnalysisDescriptionMan 객체에 GetProcessingSchedule 메시지를 보내어 입력변수에 의해 지정된 기능 블록들을 한 개씩 받아서, 새로 생성된 쓰레드(thread)에서 순서대로 Execute연산을 실행시킨다. 이러한 작업은 해당 분석 작업을 담당하고 있는 쓰레드를 제어함으로써 일시 정지, 재시작 및 중지 등의 동작을 수행할 수 있다.

파일에 저장된 신호를 분석하는 경우에는 한 화면에서 출력할 수 있는 분량의 신호를 일괄 처리한 후, 그 결과를 관찰할 수 있도록 하였다. 이미 수집된 신호를 처리하는 경우에는 현재 관찰하고 있는 부분의 전후 혹은 시작과 끝으로 이동하면서 신호를 분석하는 경우가 많다. AnalysisExecutionMan 객체는 이러한 이동을 손쉽게 할 수 있도록 GoToFirstFrame, GoToLastFrame, GoToNextFrame 및 GoToPrevFrame 등과 같은 연산자를 구비하고 있다. GoToSpecificFrame 연산자를 사용하여, 한 번에 특정한 시점으로 이동할 수도 있다. 특정한 사건들이 발생하는 시점을 쉽게 검색하기 위해 GoToFirstEvent, GoToLastEvent, GoToNextEvent, GoToPrevEvent 및 GoToSpecificEvent 연산자들을 구비하고 있다. 이 경우, AnalysisExecutionMan 객체는 찾고자 하는 사건을 검출할 때까지 이동과 분석을 계속한다.

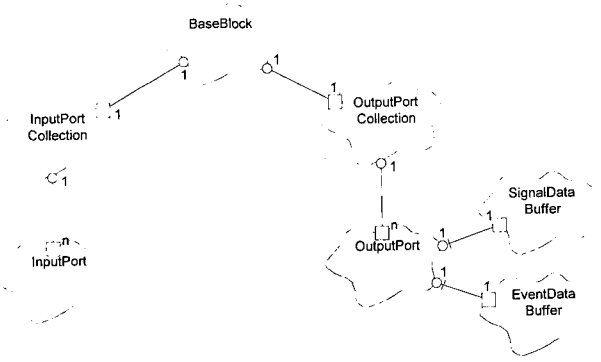

그림 6. 기능 블록 관련 클래스들의 종류와 상호 관계
Fig. 6. Relationship of classes for functional blocks

3. 신호와 사건 정보의 동시 처리를 위한 기능 블록 클래스의 구조

디지털 신호 처리 작업은 그 입력과 출력의 종류에 따라 여러 종류로 분류할 수 있다. 디지털 필터링과 같이 입력으로 받아들인 신호를 처리하여 새로운 출력 신호를 만들어내는 신호 입출력 작업, 파형 검출과 같이 신호를 입력으로 받아들여 특정 사건의 종류와 발생 시점을 추출하는 신호 입력 사건 출력 작업, 사건 정보를 처리하여 다른 사건 정보를 만들어내는 사건 입력 사건 출력 작업, 사건 정보를 처리하여 신호를 출력하는 사건 입력 신호 출력 작업 등이 있으며, 신호나 사건을 처리하여 새로운 신호와 사건을 동시에 출력하는 복합적인 신호 처리 작업들도 있다. 본 연구에서는 이러한 모든 형태의 신호 처리 작업을 수용할 수 있도록 기능 블록 클래스를 정의하였다. 그림 6은 신호 분석 환경 시스템에서 핵심적인 역할을 담당하고 있는 BaseBlock 클래스의 구조를 Booch 방법을 사용하여 나타낸 것이다.

모든 기능 블록 클래스의 바탕이 되는 BaseBlcok 클래스는 여러 개의 InputPort 클래스 객체들을 관리하는 InputPortCollection 클래스 객체와 여러 개의 OutputPort 클래스 객체들을 관리하는 OutputPortCollection 클래스 객체를 포함하고 있다. InputPort 클래스는 연결되어 있는 데이터 소스 블록 객체의 해당 OutputPort 클래스 객체에 접근할 수 있는 정보만을 보유할 수 있는 간단한 구조를 지니고 있는 반면, OuputPort 클래스는 신호 데이터와 사건 데이터를 각각 저장하는 버퍼들을 포함하고 있다. 신호 데이터 버퍼는 신호 표본들을 저장하는 배열형의 구조를 갖는데 반해, 사건 데이터 버퍼는 각 사건들의 삽입, 삭제 등을 쉽게 할 수 있도록 리스트형의 구조를 갖는다.

신호 데이터와 사건 데이터를 동기화 시키기 위해 사건 정보의 표현에 사건 발생 시점, 사건 완료 시점, 사건의 종류, 사건의 값을 나타내는 항목들을 포함하였다. 발생 시점과 완료 시점은 신호 데이터 버퍼의 인덱스로 표시하였다. 현재 보유하고 있는 신호 데이터의 이전에는 음수로, 그 이후에 발생한 경우에는 마지막 버퍼 인덱스에 초과한 만큼을 더한 수로 나타내었다. BaseBlock 클래스는 신호 데이터와 사건 데이터를 삽입하고, 입력 데이터에 대해 해당 신호 처리 알고리즘을 적용하는 인터페이스를 구비하고 있다.

4. 플러그인 확장을 위한 기능 블록 모듈의 구조

본 연구에서 개발한 신호 분석 환경 시스템은 전체 시스템의 수정 없이 원하는 신호 처리 기능을 구현한 블록 모듈만을 디렉토리에 등록함으로써 새로운 신호 처리 기능을 추가할 수 있는 플러그인 확장 기능을 구비하고 있다. 기능 블록 모듈들이 이러한 플러그인 기능을 갖기 위해서는 모듈 식별 및 객체 생성 인터페이스를 구현해야 한다. 모듈 식별 및 객체 생성 인터페이스는 해당 모듈이 구현하고 있는 기능 블록 클래스에 대한 정보를 얻을 수 있는 GetBlockClassInfo() 함수와 해당 기능 블록 객체 한 개를 새로 생성해주는 CreateBlockObject() 함수를 포함하고 있다. 기능 블록 클래스에 대한 정보는 기능 블록이 속하는 범주의 이름, 블록의 클래스 이름, 그리고 블록 클래스에 대한 설명을 나타내는 항목들로 구성된다.

기능 블록 모듈은 동적으로 링크할 수 있는 라이브러리 형태의 모듈이기 때문에 신호 분석 환경 시스템이 시동될 때, 특정 디렉토리를 검색하여 동적으로 링크할 수 있다. 신호 분석 환경 시스템은 모듈 식별 및 객체 생성 인터페이스에 포함된 GetBlockClassInfo() 함수를 호출하여 그 모듈 속에 구현되어 있는 기능 블록 클래스에 대해 알 수 있다. GetBlockClassInfo() 함수의 호출이 실패한 동적 링크 모듈은 신호 분석 환경 시스템에서 사용할 수 없는 기능 블록 모듈이다.

시스템 구현

본 연구에서 개발한 신호 분석 환경 시스템은 Windows NT환경에서 Win32 API와 C++언어를 사용하여 구현되었다. 작업 순서는 시스템 구현에 중요한 객체들을 객체 지향 분석 도구를 사용하여 모델링한 다음, C++ 클래스로 구현하였다. 기능 블록 객체들은 기본이 되는 클래스를 구현한 다음, 공통적인 속성과 메쏘드를 이 클래스에서 상속 받아 원하는 기능 블록 객체를 구현할 수 있게 하여, 코드를 최대한 재활용할 수 있도록 하였다. 각 기능 블록의 객체는 독립적인 DLL 모듈의 형태로 구현하고, 필요한 함수들만을 외부에 노출함으로써 불필요

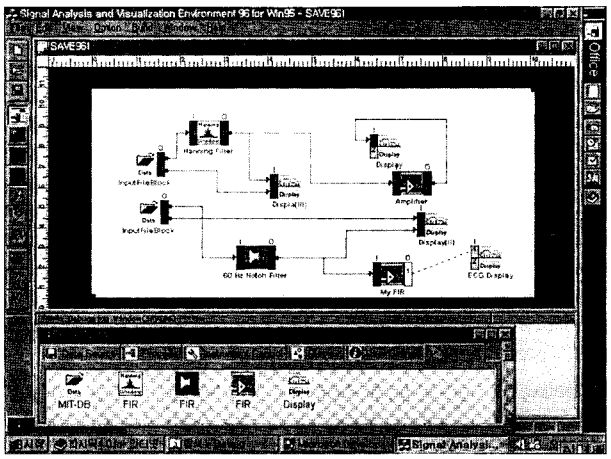

그림 7. 신호 분석 작업을 설계하는 장면
Fig. 7. Design of a signal analysis job

한 정보를 은닉하였다.

현재 구현된 데이터 소스 블록 클래스들로는 MIT에서 제정한 생체신호 MIT/BIH 형식의 생체 신호 파일을 읽을 수 있는 MITBIHFileBlock 클래스[6]와 본 연구팀이 제정하여 사용하고 있는 형식의 생체 신호 파일을 읽을 수 있는 KKUBMEFileBlock 클래스가 있다. 여러 채널의 생체 신호를 발생한 사건과 겹쳐서 한꺼번에 관찰할 수 있는 DisplayBlock 클래스가 유일하게 구현된 데이터 싱크 블록 클래스이다. 신호 처리 기능 블록 클래스로는 FIR 필터와 IIR필터가 구현되어 있으며, 고해상도 심전계에 필요한 신호 처리 기능들, 심전도의 R파 검출, 신호 평균, 형판 맞춤 등을 여러 개의 블록 클래스들로 나누어 구현하고 있다.

실 험

기본적인 기능 블록 객체들을 구현하여 신호 분석 환경을 시험하였다. 그림 7은 신호 분석 작업을 설계하고 있는 모습을 보인 것이다. 현재 사용할 수 있는 기능 블록 객체들의 아이콘들이 팔레트에 나타나 있다. 설계 작업 공간에서는 파일에 있는 신호 데이터를 읽고, 필터 기능 블록에 Notch Filtering의 효과를 낼 수 있는 필터식을 입력한 후, 화면에 출력하는 신호 분석 작업을 설계하는 모습이 나타나 있다. 그림 8은 계층적으로 등록된 기능 블록 레지스트리를 보여주고 있다. 사용자는 자신의 신호 분석에 필요한 기능 블록들을 마우스의 드래그 앤 드롭으로 선택할 수 있다.

그림 9는 설계 작업 중에 기능 블록 객체의 계수를 변경하고 있는 대화상자를 보여주고 있다. 파일 블록에 연결될 파일의 이름을 설정하고, 그 형식을 확인할 수 있다.

그림 10은 최종적으로 설계된 신호 분석 작업의 오류 유무를 검사하는 디버거(debugger)를 보여주고 있다. 기능 블록 객체들의 입력들이 모두 다른 블록의 출력에 연결되어 있는가를 검사한다.

그림 11은 최종적으로 설계된 신호 분석 작업을 실행시켜 그 결과를 화면에 출력하는 모습을 보여주고 있다. 첫 번째 채널에는 원래의 신호가 나타나 있고, 두 번째 채널에는 Notch 필터링을 거쳐서 2배로 증폭된 신호가 나타나 있다.

그림 12는 신호 파일에서 원하는 지점을 찾아가 그 곳에 있는 신호들을 관찰할 수 있게 해주는 검색 제어판의 모습을 보여주고 있다.

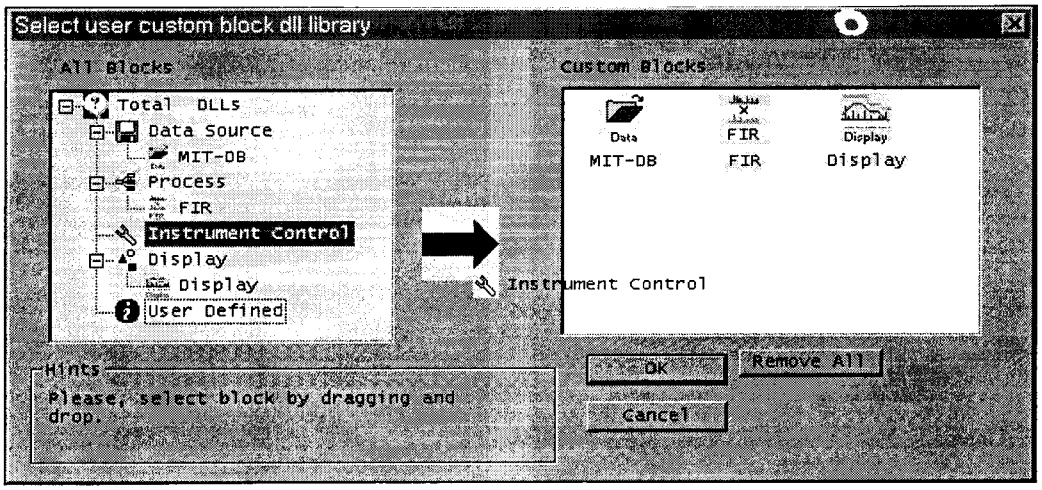

그림 8. 사용자 도구상자에 기능 블록을 등록하는 장면
Fig. 8. Block registration to a user tool box

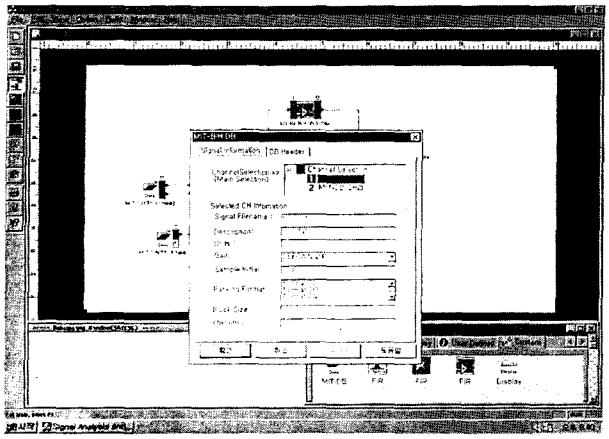

그림 9. 기능 블록 객체의 계수 변경 대화상자
Fig. 9. The parameter modification dialog box for functional block objects

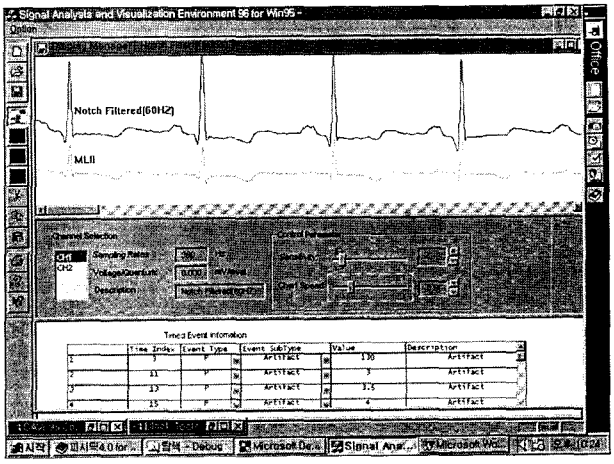

그림 11. 설계가 끝난 신호 분석 작업과 실행 결과의 화면 출력
Fig. 11. Output display of the designed signal analysis job

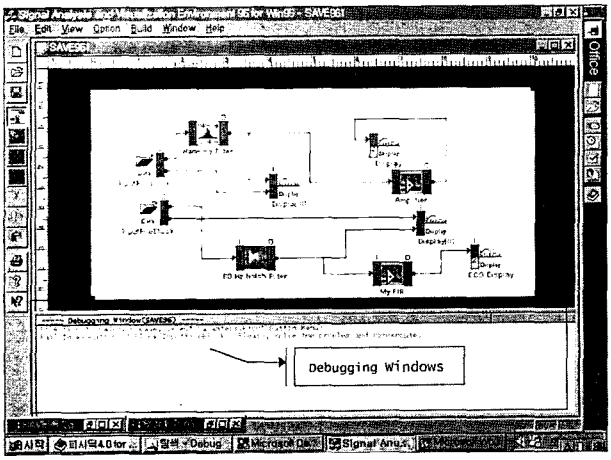

그림 10. 디버깅 화면
Fig. 10. Debugging window

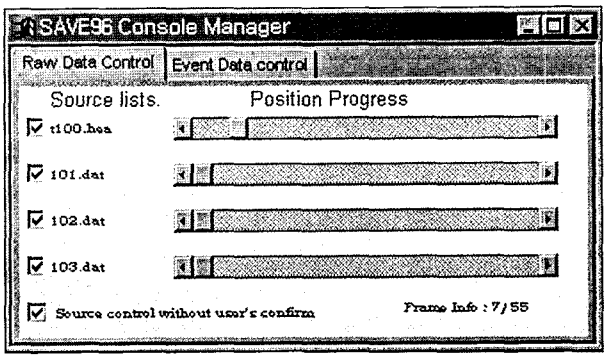

그림 12. 검색 제어판
Fig. 12. Waveform search control panel

토 의

본 연구에서 개발한 시각 프로그래밍 기능을 갖는 신호 분석 환경을 사용하면 원하는 신호 처리 작업을 빠른 시간 안에 설계하여 그 결과를 바로 확인해 볼 수가 있다. 물론, 그 신호 분석 작업에 포함된 기본적인 신호 처리 알고리즘들이 이미 기능 블록 객체의 형태로 구현되어 있다는 가정 아래서 가능한 것이다. 만약 필수적인 신호 처리 알고리즘이 구현되어 있지 않다면, 누군가가 C++ 프로그래밍 언어를 이용하여 먼저 그 알고리즘을 기능 블록의 형태로 구현해야 한다.

일반적인 프로그래밍 언어를 사용하는 경우, 신호 처리에 관한 지식 외에 프로그래밍에 필요한 지식과 시간이 많이 필요한 데 비해, 세부적인 부분까지 제어가 가능하고, 실행 속도가 빠르다는 장점이 있다. 따라서, 특정한 분야의 문제를 해결하기 위한 전용 프로그램을 작성하기에 앞서, 적용하고자 하는 해결 방안이 타당한가를 미리 확인해야 할 필요가 있는 경우이거나, 매번 다른 방법으로 신호를 분석해야 할 경우에 본 신호 분석 환경 시스템을 사용하는 것이 유리하다.

본 시스템을 사용하여 최적의 신호 분석 알고리즘을 발견한 후, 전용으로 사용할 수 있는 프로그램을 원할 경우에는 본 시스템의 기능 블록 객체들을 통합하여 바로 짧은 기간 내에 개발할 수 있다. 이 경우, 기능 블록 객체들을 표준화하기 위해 추가한 인터페이스에 의해, 신호 처리 라이브러리를 직접 사용하는 것보다 실행 속도가 다소 느려진다. 그러나, 적은 노력으로 신호 분석 환경을 통해 미리 그 결과를 확인해 보고, 바로 원하는 전용 프로그램

을 작성할 수 있다는 장점을 압도할 수는 없을 것이다.

결 론

본 연구에서는 사용자가 원하는 신호처리 작업을 시각 프로그래밍 방법으로 손쉽게 설계할 수 있고, 그 성능을 여러 종류의 시각적인 방법으로 검증할 수 있는 신호 분석 환경 시스템을 개발하였다. 신호 분석 환경은 크게 설계 작업 환경과 실행 작업 환경으로 나누어 진다. 설계 작업 환경에서는 기본적인 신호 처리, 외부 및 파일 입출력과 시각화 작업을 수행하는 기능 블록 객체들을 나타내는 아이콘들을 선택하고, 그들 사이의 입출력 관계를 연결 객체로 연결함으로써 신호 분석 작업을 설계한다. 실행 작업 환경에서는 설계 작업 환경에서 정의한 신호 분석 작업을 실행하여 그 결과를 시각화하거나 외부 혹은 파일에 출력한다. 처리 결과는 기본적으로 제공되는 시각화 기능 블록들을 사용하여 육안에 의한 관찰에 의해 검증하거나 목적에 맞는 시각화 기능 블록 객체들을 추가하여 정량적인 방법으로 검증할 수 있다. 원하는 신호의 특정 부분을 시간 정보나 혹은 그 부분에서 발생한 사건의 종류에 의해 쉽게 검색할 수 있으며, 신호 데이터와 사건 정보들을 동기화 하여 관리, 관찰할 수 있다.

모든 기능 블록 객체들은 DLL 모듈의 형태로 구현되어 있으며 공통의 인터페이스를 구비하고 있어서, 시스템의 별도 수정 없이 독립적으로 새로운 기능 블록 모듈들을 추가하여 기능을 확장할 수 있다. 최종적으로 검증된 신호 분석 작업을 위한 전용 프로그램은 이러한 모듈들을 통합하여 쉽게 구현할 수 있다.

현재는 신호 분석 환경을 시험하기 위한 간단한 기능 블록 객체들만이 구현되어 있으나, 앞으로 시간 영역 분석 및 주파수 영역 분석을 위한 기능 블록 객체들을 점차 개발할 예정이다. Distributed Component Object Model (DCOM)과 같은 분산 객체 환경을 이용하여 데이터의 수집, 분석, 저장 및 검색 작업들을 네트워크에 연결된 여러 개의 컴퓨터들에 각각 분산하여 수행할 수 있는 신호 분석 환경으로 전환하기 위한 작업이 현재 진행되고 있다.

기능 블록 객체를 구현하는데 필요한 소스 코드(source code)는 모두 공개하여, 상속 기법을 통해 자신의 목적에 맞는 알고리즘을 구현할 수 있게 할 예정이고, 신호 분석 작업의 설계와 실행을 위한 환경 시스템은 자유롭게 사용할 수 있도록 이진 파일의 형태로 원하는 사람에게 제공할 예정이다.

참 고 문 헌

1. Seung-Hun Park, J. C. Principe, J. R. Smith, and Steven A. Reid, "*TDAT-Time Domain Analysis Tool for EEG Analysis*", IEEE Trans. on Biomed. Eng., vol. 37, pp. 803-811, 1990.
2. Thomas F. Collura, Ernest C. Jacobs, David S. Braun, and Richard C. Burgess, "*Eview-A Workstation-Based Viewer for Intensive Clinical Electroencephalography*", IEEE Trans. on Biomed. Eng., vol. 40, no. 8, pp. 803-811, 1993.
3. 김형진, 박승훈, 우응제, "SiMACS에서의 생체 신호 해석을 위한 Workstation", 의용 생체 공학회 춘계학술 대회 논문집, 제16권, 제1호, pp. 60 -62, 1994.
4. 김형진, 박승훈, 우응제, "신호 분석을 위한 시각 프로그래밍 시스템", 대한 의용 생체 공학회 춘계 학술 대회 논문집, 제17권 제1호, pp. 183-185, 1995.
5. Grady Booch, "*Object-Oriented Analysis and Design*", the 2nd edition, Redwood City, California, The Benjamin/Cummings Publishing Company, Inc., 1994.
6. George B. Moody, "*ECG Database Programmer's Guide*", the 8th edition, Harvard-MIT Division of Health Sciences and Technology Biomedical Engineering Center, 1992.

=국문초록=

본 논문에서는 기본적인 신호 처리 기능 블록들을 시각 프로그래밍 기법으로 조합하여 원하는 신호 분석 작업을 수행할 수 있는 신호 분석 환경 시스템에 대해 기술하였다. 기본적인 신호 처리 알고리즘들을 기본 기능 블록의 형태로 내장하고 있으며, 사용자가 스스로 개발한 신호 처리 알고리즘들은 정해진 형식에 맞추어 기능 블록의 형태로 구현하기만 하면, 코드의 변경 없이 신호 분석 환경에 추가할 수 있다. 기본적으로 제공되는 시각화 기능 블록을 사용하여 생체신호를 비롯한 각종 1차원 신호들과 신호 처리 알고리즘에 의해 변화된 파형들을 비교, 관찰할 수 있을 뿐 아니라, 신호 검출 알고리즘에 의해 검출된 사건 정보들을 신호에 중첩하여 관찰할 수도 있다. 본 연구에서 개발된 신호 분석 환경 시스템은 객체 지향 개념에 의거하여 설계, 구현되었으며, Windows NT환경에서 동작한다.