**Proceedings of the 11th**
**KACC, October 1996**

# 효율적인 실시간 데이터 수집시스템의 설계

# Design of an Effective Real-Time Data Acquisition System

°김동욱*, 염재명*, 김대원*, 박용식*

*명지대학교 제어계측공학과 (Tel & Fax : 0335-30-6472; E-mail: dwkim@wh.myongji.ac.kr)

**Abstracts** The performance of real-time systems depends upon how well the tasks are scheduled whitin a cycle time and how fastly the response is made according to the occurrence of an external event. This paper presents the design of an effective real-time data acquisition system in order to gather the data from an automobile engine. This paper investigates an estimation and a restriction method of execution for aperiodic data. Also, the guarantee problem of real-time constraint is presented for periodic data. Through the experiments, the hard real-time guarantee problem of periodic data is studied and the damage problem of periodic data according to the increase of aperiodic tasks is analyzed.

**Keywords** real-time data acquisition system, periodic, aperiodic, guarantee, constraint

## 1. 서론

산업생산품의 품질 분석이나 개선을 위한 데이터 수집의 경우 수집된 데이터에 대한 신뢰성이 가장 중요하며, 이것은 바로 데이터 수집 시스템의 효율을 결정하는 요소이기도 하다. 신뢰성이라는 것은 수집된 데이터들의 시간성(time)과 사건성(event)에 관련된 실시간(real-time)성을 말한다. 즉, 정해진 시간에 얼마나 근접하여 데이터를 수집하는가 또는 사건 발생에 얼마나 빨리 응답할 수 있는가에 관건이 있다[3][4][9]. 그러나 수집하고자 하는 대상의 물리량 변화가 빠르거나 또는 사건발생적으로 발생되는 데이터의 수집을 사람이 직접 수행하는 경우에는 수집된 데이터들에 대한 신뢰성 부재와 자동수집기가 수행하는 만큼의 빠르고 높은 효율을 기대할 수 없다[6]. 본 논문에서는 주기적 태스크와 비주기적 태스크들이 요구할 수 있는 CPU 사용율을 예측하여 주기적 또는 비주기적으로 실행되는 태스크들의 실시간성을 만족시킬수 있도록 하며, 설계된 데이터 수집시스템에 적용하고자 한다. 논문은 2장에서 주기적 태스크와 비주기적 태스크의 실시간성 보장문제에 대한 문제의 설정을, 3장에서는 설계된 데이터 수집시스템의 구성을, 4장에서는 데이터 수집에 있어서의 실시간성 보장문제와 관련한 실험 및 결과를, 마지막으로 결론을 맺는다.

## 2. 문제의 설정

실시간 시스템은 실행의 결과가 논리적 정확성 뿐 아니라 결과가 출력되는 시간에도 영향을 받는 시스템을 말한다[3][4][9]. 즉, 주어진 시간안에 정확한 결과의 출력이 얻어져야 한다. 실시간 시스템은 태스크의 종료시한을 초과한 출력결과의 응답 시간에 따라 발생할수 있는 피해정도에 따라 하드 실시간 시스템(hard real-time system)과 소프트 실시간 시스템(soft real-time system)으로 분류할 수가 있다. 하드 실시간의 경우 정해진 시간안에 반드시 정확하고 올바른 결과값을 얻어야 하기 때문에 전체 시스템의 성능을 분석할 필요가 있다. 실시간 데이터 수집에 요구되는 시스템의 성능은 수집 데이터들에 관계하는 태스크들의 실행주기를 바탕으로 정의할수 있으며, 이를 위해 태스크들을 실행주기에 따라 주기적 태스크(periodic task)와 비주기적 태스크(aperiodic task)로 분류한다.

### *2.1 주기적 태스크들의 CPU사용율(utilization)*

식 (1)은 주기적 태스크들의 CPU사용율이다.

$$U_P = \frac{E_1}{T_1} + \frac{E_2}{T_2} + \cdots + \frac{E_n}{T_n} \quad (1)$$

여기서, $E_i$ : i 번째 태스크의 실행시간

$T_i$ : i 번째 태스크의 실행주기

n : 주기적 태스크의 수

$U_P$ : 주기적 태스크들의 CPU 사용율

### *2.2 비주기적 태스크의 CPU사용율*

데이터 수집시스템의 모든 태스크들이 시간 T를 기준으로 작동 및 정지 한다고 가정할 때 주기변화를 예측가능한 경우의 비주기적 태스크의 CPU사용율은 식 (2)와 같이 나타낼 수 있다.

$$U_A = E\left(\frac{1}{T_1} + \frac{1}{T_2} + \cdots + \frac{1}{T_N}\right) \quad (2)$$

여기서,

$T_i$ : 예측 가능한 비주기적 태스크의 i번째 실행주기,

$T = T_1 + T_2 + T_3 \ldots\ldots + T_N$

N : 태스크 실행횟수, E : 비주기적 태스크 실행시간

$E < T_{min}$, $T_{min}$ : 최소 실행 주기

$U_A$ : 비주기적 태스크의 CPU사용율

### *2.3 사건 발생적 태스크의 CPU사용율*

비주기적 태스크중에서 태스크의 실행 주기 변화를 예

측할수 없는 경우는 사건발생적 태스크(event task)로 분류할 수 있으며 이 경우에도 시간 T를 기준으로 CPU의 사용율을 정의할 수 있다. 기준시간 T 내에서 사건 발생적 태스크의 CPU사용율은 식 (3)과 같이 표현될 수 있다.

$$U_E = \frac{N \times E}{T} \qquad (3)$$

여기서, N : 시간 T에서 사건 발생적 태스크의 실행 횟수

E : 사건 발생적 태스크의 실행 시간

$U_E$ : 사건 발생적 태스크의 CPU 사용율

결과적으로 모든 태스크들의 CPU사용율은 식 (4)와 같은 조건을 만족해야만 한다.

$$U = U_P + U_A + U_E < 1 \qquad (4)$$

### *2.4 주기적 태스크의 실시간성 보장조건*

다음과 같은 6가지의 조건하에 식 (5)를 만족하면 주기적 태스크의 하드 실시간 보장이 가능하다[4].

$$U_P = \frac{E_1}{T_1} + \frac{E_2}{T_2} + \ldots + \frac{E_n}{T_n} \le n(2^{\frac{1}{n}} - 1) \qquad (5)$$

① 모든 태스크는 주기적 실행을 한다.

② 우선순위에 따른 선점형 스케쥴링 방법을 사용한다.

③ 태스크 실행 우선순위는 RMS (Rate Monotonic Scheduling)방식에 따라 설정한다.

④ $E_1 + E_2 + \ldots + E_T < T_T$

$T_T$ : Clock tick 주기, $E_i$ : 태스크 실행시간

⑤ $\frac{T_i}{T_{i-1}} \ge n$, i: 우선 순위, T: 태스크 실행 주기

⑥ 태스크들은 자원 공유나 데이터 교환을 통해 상호간에 동기화 되지 않는다.

### *2.5 비주기적 태스크의 실시간성 보장 조건*

비주기적 태스크는 시스템 인터럽트(interrupt)를 통하여 외부 사건을 받아들이며, 인터럽트 태스크의 우선순위는 모든 태스크중에서 가장 높기 때문에 외부 사건에 즉각 응답할수 있다. 결과적으로 사건 발생적 태스크의 실시간 보장성은 인터럽트 실행을 위한 인터럽트 지연(interrupt latency)시간과 인터럽트 복귀(interrupt recovery)시간에 따라 결정된다.

### *2.6 주기적 태스크의 실시간적 피해*

비주기적 태스크의 실행에 따라 발생하는 주기적 태스크의 실시간 피해를 그림 1과 같이 3가지로 분류된다.

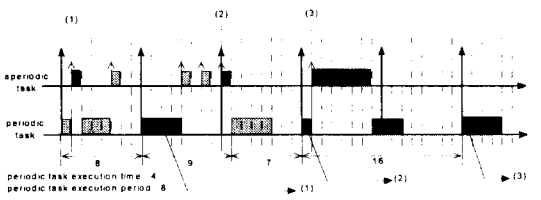


그림 1. 주기적 태스크의 실시간적 피해

Fig. 1. The real time damage of periodic task

(1) 비주기적 태스크가 주기적 태스크의 실행 중간을 선점하며, 선점된 주기적 태스크의 실행이 그 실행 주기 내에서 종료되는 경우, 주기적 태스크의 실시간성에 피해를 주지 않는다.

(2) 비주기적 태스크가 주기적 태스크의 실행 바로 이전에 그 실행을 선점하는 경우, 주기적 태스크의 실시간적 피해는 선점된 시간만큼 발생된다.

(3) 비주기적 태스크가 주기적 태스크의 실행 중간을 선점하며, 선점된 주기적 태스크의 실행이 그 실행 주기 내에서 종료되지 못하는 경우, 식 (6)과 같은 실시간적 피해가 발생된다.

$$\frac{\text{초과된 clock tick}}{\text{주기적 태스크의 실행 주기}} \times 100\ (\%) \qquad (6)$$

### *2.7 비주기적 태스크 실행 증가에 의한 주기적 태스크의 실시간적 피해*

비주기적 태스크의 실행이 증가함으로써 발생되는 주기적 태스크의 실시간적 피해를 줄이기 위해 비주기적 태스크에 대한 제약 조건을 제시한다.

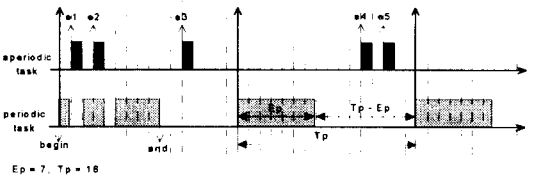


그림 2. 비주기적 태스크에 의해 선점되는 주기적 태스크

Fig. 2. A periodic task preempted by aperiodic task

비주기적 태스크의 실행 주기 변화율이 일정한 비율로 변화한다고 할 때 주기적 태스크의 실행 주기를 $T_p$라고 한다면 주기적 태스크의 실시간 보장조건은 식 (7)과 같다.

$$t_{end} - t_{begin} \le T_p \qquad (7)$$

여기서, $t_{begin}$은 주기적 태스크의 실행 시작시간을, $t_{end}$는 종료시간을 나타내며, 식 (7)은 식 (8)과 같이 표현된다.

$$t_{end} - t_{begin} = E_p + E_{pr} \qquad (8)$$

또한, $E_{pr}$은 주기적 태스크 실행이 비주기적 태스크에 의해 선점되는 시간을 의미하며 식 (9)와 같이 표현된다.

$$E_{pr} = \sum_{t=begin}^{end} (E_a^t \times e_t) \qquad (9)$$

이때, 시간 t에서 비주기적 태스크의 실행이 발생되면 $e_t$는 1의 값이 되며, 태스크 실행시작이 아닌경우에는 0의 값이 된다. 여기서, $E_a^t$ 는 시간 t에서의 비주기적 태스크의 실행시간을 의미한다. 식 (7), (8), (9)로부터

$$\sum_{t=begin}^{end} (E_a^t \times e_t) \le (T_p - E_p) \qquad (10)$$

만일, $E_a^t$ 가 상수라면,

$$\sum_{t=begin}^{end} e_t \le \frac{(T_p - E_p)}{E_a^t} \qquad (11)$$

식 (11)에서, $\sum_{t=begin}^{end} e_t$ 는 주기적 태스크의 실행이 그 실행 주기 안에서 사건 발생적 태스크에 의해 선점될수 있는 횟수로서 전체 시스템의 동작시간 T를 고려한 최악의 경우, 식 (12)와 같은 표현이 가능하다.

$$N = MAX(\sum_{t=0}^{T} e_t) = T + 1 \qquad (12)$$

결과적으로 시스템의 동작시간 T사이에서 수용할수 있는 비주기적 태스크의 총 실행 횟수 N은 식 (13)과 같이 나타나며, N과 T를 이용하여 주기적 태스크의 실시간 보장하에 시스템에 최대로 적용할 수 있는 비주기적 태스크

의 실행 주기(period)를 제약할 수 있다. 식 (13)에서 ⌈ ⌉ 는 ceiling function이다.

$$N \le \lceil \frac{T - E_p^T}{E_a^T} \rceil - 1 \qquad (13)$$

여기서, $E_a^T$는 시간 T내에서 비주기적 태스크의 총 실행시간을, $E_p^T$는 주기적 태스크의 총 실행시간을 나타낸다.

## 3. 시스템의 구성

본 논문에서 구성된 데이터 수집 시스템은 전체적으로 외부의 신호를 수집하고 이를 가공, 저장, 전송 및 모니터링을 수행한다. 임의 타겟(target)으로 설정한 온도 변화 데이터는 실제의 모델을 설치하고 실험하기가 곤란하기 때문에 온도 센서 시뮬레이터로서 그 역할을 대신하도록 한다. 데이터 수집장치에서 수집되고 저장된 데이터들은 통신라인을 통하여 호스트(host) 컴퓨터로 전송되어 좀더 진보된 형태의 데이터 가공이나 고수준 사용자 인터페이스(high level user interface)를 제공할 수 있게 한다. 다음 그림 3은 전체 시스템의 구성을 보여준다.

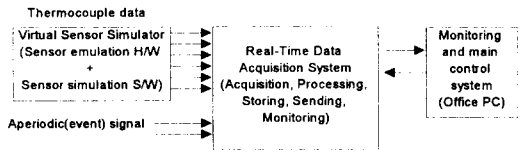


그림 3. 전체 시스템의 구성도
Fig. 3. The block diagram of the entire system

### *3.1 실시간 데이터 수집 시스템*

실시간 데이터 수집시스템의 구성은 외부 신호를 수집할 수 있는 하드웨어와 이를 실시간으로 운영하는 실시간 소프트웨어(real-time software)로 구성되며, 본 논문에서 다루고 있는 부분이다. 외부 신호를 입력하는 하드웨어는 analog input board, analog output board, asynchronous(event) input board, counter input board, digital input/output board의 5가지로 분류하여 제작하였다. 각각의 하드웨어를 특성별로 분리하여 설계, 제작하는 이유는 쉬운 확장성과 시스템 재설계의 용이성을 고려하기 때문이다. 실시간 데이터 수집 시스템 소프트웨어는 프로그램의 확장성과 재사용성을 고려하여 하드웨어 모듈의 추가 및 삭제에 유연하게 대처할수 있도록 5가지 하드웨어 모듈에 바탕을 둔 구조적, 모듈적 프로그램으로 구성하였다. 데이터 수집 시스템의 실시간 문제와 관련하여 $\mu$C/OS(micro C / O S)라는 공개 RTOS를 사용한다. 이것은 선점형 멀티 태스킹하에 62개의 태스크와 62개의 우선순위를 설정할수 있으며, 커널(kernel)소스가 공개되어 있어 다른 CPU로의 이식성이 유리한 장점이 있다[3][4].

### *3.2 센서 시뮬레이터*

센서의 출력은 열전쌍(thermocouple)을 대상으로 설정하였으며, 시뮬레이터 소프트웨어는 $\mu$C/OS를 사용한 실시간 소프트웨어를, 센서 전압 출력은 각 채널마다 12비트의 분해능을 갖는 16채널 analog output board를 사용한다. 시간에 따른 온도센서값의 출력은 임의의 범위 안에서 BETA 분포에 따른 난수값(random variable)을 사용한다.

### *3.3 모니터링 시스템*

모니터링 시스템에서는 실시간 데이터 수집 시스템에서 수집된 데이터들을 GUI(Graphical User Interface)환경하에서 분석 및 모니터링할 수 있도록 하며, 또한 RS-485와 같은 통신라인을 통하여 다수의 데이터 수집 시스템과 연결할 수도 있다. 모니터링 시스템 소프트웨어는 엔드유저(end user)에 의해 변형이나 설계가 가능한 상용화된 데이터 수집 소프트웨어를 사용하거나[2][8] 또는 특정 모델에 의존적인 소프트웨어를 작성하는 경우로 나눌 수 있다.

## 4. 실험 및 결과

### *4.1 실험 환경 구성*

실험은 주기적 태스크와 비주기적 태스크들의 실시간성 보장 문제를 다루는 것을 목적으로, 주기적 태스크는 analog input board를, 비주기적 태스크는 시스템 인터럽트를 사용하며 소프트웨어도 이에 맞게 간략화 하였다.

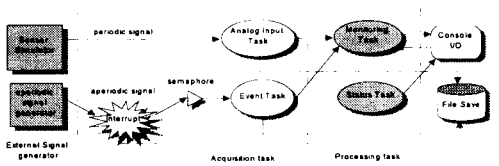


그림 4. 실험을 위한 시스템 구성도
Fig. 4. The block diagram for experimental system

### *4.2 주기적 태스크들의 CPU사용율 측정*

표 1과 같은 조건에서 주기적 태스크의 실행주기를 변화 시켰을 때 CPU사용율을 예측한값과 측정한값을 표 2에 나타 낸다.

표 1. 태스크 실행 시간
Table 1. Task execution time

| 태스크 이름 | 실행시간 | delay tick |
|---|---|---|
| Analog Input 태스크 | 2370 $\mu$Sec | 2(105094 $\mu$Sec) |
| Monitoring 태스크 | 32178 $\mu$Sec | 가변 |
| Status 태스크 | 234 $\mu$Sec | 18(945846 $\mu$Sec) |
| Event 태스크 | 139 $\mu$Sec | 가변 |

표 2. 주기적 태스크들의 CPU 사용율
Table 2. CPU utilization for periodic tasks

| Monitoring 태스크 delay tick | 식 (1)에 의한 계산값 | 프로그램 실행시 측정값 |
|---|---|---|
| 2 | 32% | 31.72% |
| 6 | 12.4% | 12.01% |
| 18 | 5.6% | 5.48% |

### *4.3 태스크의 실시간 보장성에 관한 실험*

그림 5와 표3은 주기적 태스크들만 실행하였을 때의 실시간 보장성을 확인하는 실험이다. 데이터 수집 시스템 및 시뮬레이터 각각의 입출력 샘플링 시간은 2 clock tick(약 110ms)으로 동일하며 그래프의 x축을 의미한다. 표에서 simulator output과 acquisition input행에 기록된 값은 A/D변환기와 D/A변환기에 직접 적용되는 비트값(bit value)으로 그래프의 y축을 의미한다. 시뮬레이터에서는

610℃(bit value = 1750) ~ 1350℃(bit value = 3500) 까지의 K-type 열전쌍 온도에 대비되는 전압을 출력하였다.

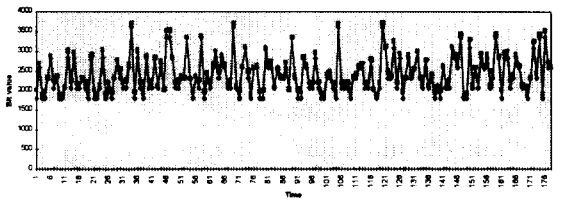

그림 5. 하드 실시간으로 수집된 데이터 그래프
Fig. 5. The graph of hard real-time acquisition data

표 3. 하드 실시간으로 수집된 데이터 표
Table 3. Hard real-time acquisition data

| Time [분:초:백분초] | Acquisition Input | Time [분:초:백분초] | Simulator Output |
|---|---|---|---|
| [10:33:94] | 2761 | [10:33:94] | 2579 |
| [10:34:05] | 2546 | [10:34:05] | 2320 |
| [10:34:16] | 2216 | [10:34:16] | 2063 |

그림 6과 표 4는 주기적 태스크의 하드실시간 보장 조건 ③을 위반한 경우에 대한 데이터 그래프와 표이다. 실험은 실행주기가 2 clock tick인 analog input태스크의 우선순위보다 실행주기가 10 clock tick인 monitoring task의 우선순위를 높게 설정한 것이며, 표 4의 acquisition input 행에서 굵게 표시된 시간이 데이터 수집시스템의 실시간적 피해를 보여준다.

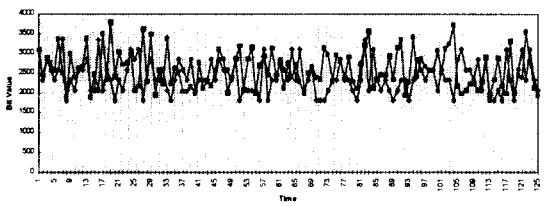

그림 6. 소프트 실시간으로 수집된 데이터 그래프
Fig. 6. The graph of soft real-time acquisition data

표 4. 소프트 실시간으로 수집된 데이터 표
Table 4. Soft real-time acquisition data

| Time [분:초:백분초] | Acquisition Input | Time [분:초:백분초] | Simulator Output |
|---|---|---|---|
| [55:48:18] | 2328 | [55:48:18] | 2546 |
| **[55:48:89]** | 2586 | [55:48:29] | 3363 |
| [55:49:00] | 3359 | [55:48:40] | 2491 |
| [55:49:11] | 1811 | [55:48:51] | 1904 |
| [55:49:22] | 2328 | [55:48:62] | 2992 |
| [55:49:33] | 2070 | [55:48:73] | 2380 |
| **[55:50:05]** | 2585 | [55:48:84] | 2628 |
| [55:50:16] | 2586 | [55:48:95] | 2659 |

비주기적 태스크의 실행에 따른 주기적 태스크의 실시간적 피해를 확인하기 위하여 다음과 같은 조건을 설정한다. 비주기적 태스크의 실행시간 $E_a^T$는 139[ $\mu$Sec], 시스템 동작시간 T는 3[Sec], 주기적 태스크의 실행시간 $E_p^T = T \times U_p = 3 \times 0.32 = 0.96$[Sec], 식 (13)으로부터 N≤14676의 조건이 성립된다. 즉, 시스템 동작시간 3초 내에서 주기적 태스크의 CPU 사용율 $U_p$와 함께 최대로 실행할수 있는 비주기적 태스크의 실행 회수는 14,676회 이다. 또한 식 (2)의 조건으로부터

$f_{max} \le \frac{1}{E_a^T} = \frac{1}{139uSec} = 7194$ Hz을 만족해야 한다. 예로서 스텝모터 가속 펄스(pulse)공식의 경우 시간 T내에서 펄스수 N일 때 최대 주파수는

$f_{max} = \frac{2 \times N}{T}$ 이다[5].

따라서 $f_{max} = \frac{2 \times 14676}{3} = 9784$ Hz이 되며, 결과적으로 주기적 태스크들의 CPU사용율 32%, 시스템의 동작시간 3초 내에서 비주기적 태스크의 최소 실행 주기는 $t_{min}$ = 139[ $\mu$Sec](7194 Hz)가 되며 비주기적 태스크의 최대 실행회수 N은 14,675회 가능하다.

## 5. 결 론

논문에서는 주기적, 비주기적 태스크의 실시간성 보장 조건을 제시하기 위하여 비주기적 태스크를 시스템의 동작시간 T를 기준으로 예측가능한 경우와 예측 불가능한 경우로 분류하여 시스템 분석에 사용하였고, 실험을 통하여 주기적 태스크들의 실시간 보장성을 확인하였으며, 비주기적 태스크들의 실행 조건을 제시하였다. 결과적으로 주기적 또는 비주기적으로 발생되는 외부 데이터 수집을 위한 실시간 데이터 수집시스템의 신뢰성을 부여하는 효율적인 데이터 수집시스템 설계 방법을 제시하였다. 앞으로 실제 환경인 자동차 엔진에 접속하여 실시간 데이터 수집실험이 수반되어야 하겠다.

## 참고문헌


[1] H. G. Rotithor, "A High-Performance Pipelined Architecture for Measurement and Monitoring of Multiple Sensor Signals," *IEEE Trans. on instrumentation and measurement*, Vol. 41, No. 6, pp.808-814, Dec. 1992

[2] Howard Austerlitz, *Data Acquisition Techniques Using Personal Computers*, Academic Press, USA, 1991

[3] Jean J. Labrosse, *μC/OS The Real-Time Kernel*, R&D Publications, USA, 1992

[4] Jean J. Labrosse, *Embedded Systems Building Blocks*, R&D Publications, USA, 1995

[5] KATSUHIKO OGATA, *SYSTEM DYNAMICS*, Prentice Hall, U.S.A, 1992

[6] Pasquale Daponte, Libero Nigro, Francesco Tisato, "Object-Oriented Design of Measurement Systems," *IEEE Trans. on instrumentation and measurement*, Vol. 41, No. 6, pp.874-880, Dec. 1992

[7] Stewart V. Hoover, Ronald F. Perry, *SIMULATION*, Addison-Wesley, USA, 1989

[8] *INSTRUMENTATION REFERENCE AND CATALOGUE*, NATIONAL INSTRUMENTS, 1996

[9] *VRTX/Spectra Training Manual*, 한국 마이크로텍

[10] *RTKernel 4.5 User's Manual*, ON Time, 1994