한국산학기술학회논문지
Vol. 10, No. 10, pp. 2760-2765, 2009

개선된 동적 퀀텀 크기 Pfair 스케줄링의 구현

김남진[1], 김인국[1*]
[1]단국대학교 컴퓨터학부

An Implementation of Improved Dynamic Quantum-Size Pfair Scheduling

Nam-Jin Kim[1] and In-Guk Kim[1*]
[1]**Department of Computer Science, Dankook University**

요 약 다중 프로세서 환경에서 경성 실시간 태스크 집합의 스케줄링 문제를 해결하는 Pfair 스케줄링 알고리즘은 고정된 퀀텀 크기를 기반으로 한다. 최근 mode change 환경에서 최대 퀀텀 크기를 동적으로 결정하는 방법이 제안되었는데, 이 방법에서는 태스크들의 주기가 감소되는 경우만을 다루고 있다. 본 논문에서는 태스크들의 주기가 증가되는 경우까지도 고려하여 최대 퀀텀 크기를 동적으로 결정하는 개선된 방법을 제안하였고 모의실험에서 이를 구현하여 효율성을 입증하였다.

Abstract Pfair scheduling algorithm, which is an optimal scheduling algorithm in the hard real-time multiprocessor environments, is based on the fixed quantum size. Recently, several methods that determine the maximum quantum size dynamically were proposed in the mode change environments. But these methods considered the case in which the period of a task can only be decreased. In this paper, we consider the case in which the period of a task can be decreased or increased, and propose an improved method that determine the maximum quantum size dynamically in the mode change environments. A simulation shows that the proposed method is effective.

Key Words : Real-time Scheduling, Multiprocessor Scheduling, Pfair Scheduling

1. 서론

컴퓨터 시스템에서 수행되는 각 작업은 그것에 의해 의도된 논리적인 흐름에 맞게 수행되어 올바른 결과를 산출해 내는 것을 기본 목표로 한다. 실시간 시스템에서는 이러한 논리적인 올바름에 처리시한의 제한(deadline, 이하 종료시한)을 지키는 것이 부가적으로 요구되며 만약 주어진 종료시한이내에 해당 작업을 처리하지 못하는 경우에는 시스템에 심각한 문제를 초래할 수도 있다.

따라서 실시간 시스템에서 스케줄링은 매우 중요한 의미를 가지고 있으며 기존의 범용 시스템의 경우와는 방법이 매우 다르다. 이는 각 태스크에게 종료시한이 있고 주어진 종료시한이내에 태스크가 수행 완료되어야만 실행의 의미가 있다는 점으로부터 기인한다.

단일 프로세서 환경에서는 Liu와 Layland가 제안한 Rate-Monotonic Scheduling(RMS) 알고리즘과 Earliest Deadline Scheduling(EDS) 알고리즘이 최적의 알고리즘으로 증명되었다[3]. 그러나 RMS나 EDS는 다중프로세서 환경에서는 최적의 알고리즘이 아니다.

최근 Baruah 등은 다중 프로세서 환경에서 주기적 경성 실시간 태스크들에 대한 최적의 스케줄링 알고리즘인 Pfair scheduling(PF)[10] 알고리즘과 이것을 효율적으로 개선한 PD[9] 알고리즘을 제안하였다. 이 논문을 통해 Baruah 등은 프로세서의 수가 M일 때 태스크 집합이 스케줄링 가능하기 위한 필요충분조건이 다음과 같음을 증명하였다. 여기서, $T.e$와 $T.p$는 태스크 집합 τ에 속한

이 연구는 2008년도 단국대학교 대학연구비의 지원으로 연구되었음.
*교신저자 : 김인국(igkim@dankook.ac.kr)
접수일 09년 08월 18일 수정일 09년 10월 09일 게재확정일 09년 10월 14일

태스크 T의 실행 요구 시간과 주기를 나타낸다.

$$\sum_{T\in\tau}\frac{T.e}{T.p} \leqq M \tag{1}$$

이 밖에도 Pfair를 기반으로 하는 다수의 알고리즘들이 제안되었지만, 이러한 퀀텀 기반의 스케줄링 알고리즘들에서 주기 및 실행 요구 시간은 퀀텀 크기의 배수로 가정하고 있다[4-8]. 그러나 태스크 집합의 대주기에서 태스크 집합이 새로운 태스크 집합으로 변경(mode change)되는 상황[1]이나 동적 태스크 집합에 대해서는 스케줄링의 효율성 또는 스케줄링 가능성을 위해 각 태스크의 실행 요구 시간 및 주기에 따른 퀀텀의 크기 변경이 요구될 수 있다. 예를 들어, 변경된 태스크 집합에 대한 모든 태스크들의 주기 및 실행 요구 시간이 퀀텀 크기의 정수 배수 n일 경우 퀀텀의 크기를 n으로 증가시킬 수 있다.

퀀텀 크기의 증가는 문맥 교환과 캐시 재적재 시간과 같은 오버헤드를 감소시킴으로서 전반적인 시스템 효율을 향상시킬 수 있다. 따라서 주어진 태스크 집합에 대해 최적의 퀀텀 크기는 스케줄링이 실패하지 않는 한도내에서 최대로 결정하는 문제이다. 최근 mode change 환경에서 태스크 집합이 스케줄링 가능하면서 퀀텀 크기를 최대로 결정하기 위한 방법이 제안된 바 있다[2]. 그러나 이 논문에서 제안된 방법에서는 mode change 될 때 각 태스크의 주기가 같거나 감소되는 경우만을 다루고 있는데, 보다 일반적인 경우가 되기 위해서는 태스크의 주기가 증가되는 경우도 포함되어야 할 것이다. 이에 본 논문에서는 태스크의 주기가 같거나 감소되는 경우뿐 만아니라 증가되는 경우도 고려하여 최적 퀀텀 크기를 결정하는 새로운 방법을 제안하고자 한다.

2. 스케줄링 환경 및 태스크 모델

본 논문에서 고려하고 있는 스케줄링 방법은 전역 스케줄링을 기반으로 하고 있으며, 프로세서간 이동이 허용되는 태스크 집합은 구 스케줄(old schedule)의 대주기(major period) 마지막에서 모드가 변경됨을 가정한다. 대주기는 태스크들의 주기의 최소공배수로서 모든 태스크들은 대주기의 시작점에서 동시에 시작될 수 있다고 가정한다.

태스크들은 주기의 변경 허용성에 따라 두 가지로 분류될 수 있다. 이들은 퀀텀 크기의 변경이후에도 원래의 주기를 엄격히 지켜야 하는 주기 불변 태스크와 주기의 변경이 허용되는 주기 가변 태스크이다. 본 논문에서 제시되는 스케줄링 방법은 주기 가변 태스크들을 대상으로 하며 주기 가변 태스크는 각 작업의 발생 간격이 모드 변경 시점 이전보다 짧아지거나 길어질 수 있는 태스크이다.

2장의 나머지 부분에서는 앞으로 사용하게 될 기호 및 관련 식들에 대해 정의하기로 한다. 각 태스크 T는 주기 p와 실행요구 시간 e를 속성으로 갖는다. $q_{\min}$은 시스템에서 표현할 수 있는 최소 퀀텀 크기로서 단위 시간 1을 의미한다. q_c는 현재 적용 중인 퀀텀 크기이고, q_n은 새롭게 선택되는 퀀텀의 크기로서 q_n이 결정되면 $q_c = q_n$이 된다.

또한, e_c와 p_c는 q_c에, e_n과 p_n은 q_n에 비례해서 정의되는 실행 요구 시간과 주기를 나타낸다. 이때, 퀀텀 크기의 변경에 따라 태스크의 주기는 증가되거나 감소될 수 있으며, 실행요구 시간은 감소되지 않아야 하므로 e_n과 p_n은 다음과 같이 표현될 수 있다.

$$e_n = \left\lceil \frac{e}{q_n} \right\rceil \tag{2}$$

$$p_n = \begin{cases} \left\lfloor \frac{p}{q_n} \right\rfloor \text{ or } \left\lceil \frac{p}{q_n} \right\rceil & (\text{if} \quad q_n < p) \\ 1 & (\text{if} \quad q_n \geqq p) \end{cases} \tag{3}$$

따라서 w를 $q_{\min}$에 의해 계산되는 태스크의 프로세서 이용률(utilization)이라 하면, q_n에 의해 증가된 태스크의 이용률 w'은 다음과 같이 표현된다.

$$w' = \frac{e_n}{p_n} = \begin{cases} \dfrac{\left\lceil \frac{e}{q_n} \right\rceil}{\left\lceil \frac{p}{q_n} \right\rceil} \text{ or } \dfrac{\left\lceil \frac{e}{q_n} \right\rceil}{\left\lceil \frac{p}{q_n} \right\rceil} & (\text{if } q_n < p) \\ 1 & (\text{if } q_n \geqq p) \end{cases} \tag{4}$$

그러므로 $T.w'$를 태스크 집합 τ에 속한 각 태스크 T의 이용률이라 하면 τ에 속한 모든 태스크들의 이용률의 합 U는 다음과 같이 구할 수 있다.

$$U = \sum_{T \in \tau} T.w' \quad (5)$$

이후에는 특별한 언급이 없는 한 선택되는 퀀텀 크기로 q_c와 q_n대신 기호 Q를 사용하기로 한다. 또한 $p_{\max}$와 $p_{\min}$은 태스크들에 대한 주기의 최대값과 최소값을 의미한다.

3. 최적 퀀텀 크기를 결정하기 위한 방법

3.1 도달점

주어진 태스크 집합에서 퀀텀 크기가 증가할 때 U는 단조 증가하지는 않는다. 그러나 Q를 계속 증가시키다 보면 어느 시점부터는 w'가 계속 1의 값을 유지하게 되는데, 이렇게 w'가 연속적으로 1이 되는 최초의 Q를 실제 도달점 RRP라 하고, 실제 도달점을 포함한 이후의 Q값들을 도달 구간이라고 한다.

실제 도달점을 구하기 위해서는 $p_{\max}-1$로부터 Q값들을 1만큼씩 감소시키면서 이용률이 1보다 작은 값이 나오는 지를 반복적으로 찾아야 하는데, 본 논문에서는 주기가 증가하지 않는 태스크들 뿐 만아니라, 주기가 증가하는 태스크들까지도 포함된 태스크 집합에 대하여 최적의 퀀텀 크기를 구하기 위한 다항식 함수를 제안하고자 한다.

[2]에서는 주기가 증가하지 않는 태스크들의 도달점을 구하기 위한 다항식 함수를 다음과 같이 제시하였다.

[정리 1]

주기가 증가하지 않는 태스크의 도달 함수 $Reach(p,\ e)$는 다음과 같이 정의한다.

$$Reach(p,\ e) = \begin{cases} \left\lfloor \frac{p}{2} \right\rfloor + 1 & \left(\text{if} \quad \frac{e}{p} \leq \frac{1}{2}\right) \\ \left\lfloor \frac{p}{3} \right\rfloor + 1 & \left(\text{if} \quad \frac{e}{p} > \frac{1}{2}\right) \end{cases} \quad (6)$$

이때, $Q \geq Reach(p,\ e)$이면

임의의 p와 e에 대해 $w' = \dfrac{\left\lceil \frac{e}{Q} \right\rceil}{\left\lfloor \frac{p}{Q} \right\rfloor} \geq 1$이다.

본 논문에서는 주기가 증가하는 태스크들의 도달점을 구하기 위한 다항식 함수를 다음과 같이 제시한다.

[정리 2]

주기가 증가하는 태스크의 도달 함수 $Reach(p,\ e)$는 다음과 같이 정의한다.

$$Reach(p,\ e) = p \quad (7)$$

이때, $Q \geq Reach(p,\ e)$이면

임의의 p와 e에 대해 $w' = \dfrac{\left\lceil \frac{e}{Q} \right\rceil}{\left\lfloor \frac{p}{Q} \right\rfloor} \geq 1$이다.

[증명]

$Q \geq Reach(p,e)$이면 $Q \geq p$이고, 따라서 $\left\lceil \frac{e}{Q} \right\rceil$와 $\left\lceil \frac{p}{Q} \right\rceil$는 각각 1이다. 따라서 $w' = \frac{1}{1} \geq 1$이다.

3.2 예제

주어진 태스크 집합

$\tau = \left\{ T_1 = \frac{7}{41}, T_2 = \frac{2}{4}, T_3 = \frac{29}{48}, T_4 = \frac{8}{39}, T_5 = \frac{15}{22} \right\}$

을 스케줄링 할 때 프로세서의 수가 M이라 가정하자. 만일 M개의 태스크가 도달 구간에 있다면, 이미 M개의 태스크에 의해 $U = M$이 되었고 나머지 태스크들의 이용률의 합은 적어도 0보다 크므로 스케줄링은 실패하게 된다.

따라서 최적의 Q값은 적어도 M번째 태스크의 도달점보다 작은 구간에서 구해진다. 이를 위해 도달 순위에 따라 도달점을 기억하는 도달 순위 리스트 $Rank[n]$을 생성하는데 각 태스크의 도달점을 오름차순으로 저장한다. 실제 도달점보다 작은 값으로부터 Q값을 찾아야 하므로 리스트에는 $Reach(p,e)-1$의 값이 저장된다.

이제 주어진 태스크 집합에 대하여 다음의 경우들을 고려해 보자. 첫 번째는 τ에 속한 모든 태스크들이 주기가 증가하지 않는 태스크들이라고 가정한 경우이고, 두 번째는 T_1, T_3, T_4는 주기가 증가하지 않는 태스크들이고 T_2와 T_5는 주기가 증가하는 태스크들이라 가정한 경우이며, 세 번째는 τ에 속한 모든 태스크들이 주기가 증가하는 태스크들이라 가정한 경우이다.

3.2.1 첫 번째 경우

태스크 집합 τ에 대한 도달 순위 리스트는 표 1과 같다. 이를 기반으로 Q를 결정하기 위한 방법은 [2]에서 제시하였다.

[표 1] τ에 대한 도달 순위 리스트(첫 번째 경우)

i	0	1	2	3	4
Rank[i]	2	7	16	19	20
Task 번호	T_2	T_5	T_3	T_4	T_1

[알고리즘 1]에서는 주어진 M에 따라 도달 순위 리스트에서 $Rank[M-1]$을 찾고, 이 경우 이용률의 합이 M보다 작거나 같은 가를 확인해 본다. 그렇다면 주어진 태스크 집합은 Pfair 스케줄링 알고리즘에 의해 스케줄링 가능한 것이고 최적의 퀀텀 크기는 $Rank[M-1]$이 된다.

이용률의 합이 M보다 크다면 도달 순위 리스트에서 $Rank[M-2]$를 찾고 이 경우의 이용률의 합이 M보다 작거나 같은 가를 확인해 본다. 그렇다면 주어진 태스크 집합은 스케줄링 가능한 것이고, 이 경우의 최적 퀀텀 크기는 $Rank[M-2]$가 된다.

이러한 과정은 스케줄링이 가능해 질 때까지 반복 실행되는데 경우에 따라서는 Q의 값이 최적의 값보다 작은 것으로 구해질 수도 있다. 이 경우에는 $Rank[M-1]$로부터 퀀텀의 크기를 1만큼씩 감소시키면서 Q를 찾으면 된다. 이상에서 소개된 Q를 구하는 함수는 두 번째 경우와 세 번째 경우에도 그대로 적용될 수 있다.

```
int FindQ() {
    int Q = 1;
    int i;
    for(i=M-1; i>=0; i--) {
        if(calculate_U(ts, Rank[i]) <= M) {
            Q = Rank[i];
            break;
        }
    }
    if(Q==1) {
        for(i=Rank[M-1]; i>0; i--){
            if(calculate_U(ts, i) <= M) {

                Q = i;
                break;
            }
        }
    }
    return Q;
}
```

[알고리즘 1] 최적 퀀텀 크기 Q를 결정하는 함수

3.2.2 두 번째 경우

두 번째 경우는 τ에 주기가 증가하지 않는 태스크들과 주기가 증가하는 태스크들이 모두 포함된 경우이다. 이 경우의 도달 순위 리스트는 표 2와 같다.

태스크 집합 τ에 대해 $M=4$일 때 $Rank[3]=20$이고, 이때 $U \leq 4$이므로 태스크 집합은 Pfair 스케줄링 알고리즘에 의해 스케줄링 가능하고 최적의 퀀텀 크기는 20이 된다. 이를 첫 번째 경우와 비교해 보면 퀀텀 크기가 19에서 20으로 1만큼 증가하여 개선되었음을 볼 수 있다.

그러나 $M=3$이면 $Rank[2]=19$와 $Rank[1]=16$에서 각각 스케줄링이 실패하고 $Rank[0]=3$에서 스케줄링 가능하게 된다. 이는 실제 최적의 퀀텀 크기인 5와는 2만큼의 차이가 발생한다. 이러한 경우에는 $Rank[M-1]$로부터 퀀텀의 크기를 1만큼씩 감소시키면서 최적의 Q를 구할 수 있다.

[표 2] τ에 대한 도달 순위 리스트(두 번째 경우)

i	0	1	2	3	4
Rank[i]	3	16	19	20	22
Task 번호	T_2	T_3	T_4	T_1	T_5

3.2.3 세 번째 경우

세 번째는 τ에 속한 모든 태스크들이 주기가 증가하는 태스크들인 경우이다. 이 경우의 도달 순위 리스트는 표 3과 같다. $M=4$일 때 $Rank[3]=40$이고 이 경우에 $U \leq 4$이므로 태스크 집합은 Pfair 스케줄링 알고리즘에 의해 스케줄링 가능하고 최적의 퀀텀 크기는 40이 된다. 이는 첫 번째나 두 번째 경우에 비해 현저하게 개선되었음을 볼 수 있다.

$M=3$이면 $Rank[2]=38$에서는 스케줄링이 실패하나 $Rank[1]=21$에서는 $U=2.8975$로 스케줄링이 가능하여 이 부분에서도 첫 번째나 두 번째 경우에 비해 개선되었음을 볼 수 있다.

[표 3] τ에 대한 도달 순위 리스트(세 번째 경우)

i	0	1	2	3	4
Rank[i]	3	21	38	40	47
Task 번호	T_2	T_5	T_4	T_1	T_3

4. 구현 및 분석

본 장에서는 앞에서 살펴본 세 가지 경우에서 구한 최

적의 Q값과 스케줄링 가능성 여부를 비교 분석하여 본다. 본 실험은 Windows XP가 설치된 X-86기반의 플랫폼에서 C 언어 프로그램으로 구현하였으며 최대 주기가 1,000이고 전체 이용률이 4를 넘지 않도록 무작위로 생성된 100,000개의 태스크 집합을 대상으로 실험하였다.

표 4는 실험의 결과를 보여준다. 여기서 평균 퀀텀 크기는 실험 대상이 된 태스크 집합들의 최적 퀀텀 크기의 평균이며 $M=4$인 경우의 실패 횟수는 프로세서의 수가 4인 경우에 실제로는 1보다 큰 Q값이 존재함에도 불구하고 그 값을 찾지 못해 Q값이 1로 결정된 총 횟수를 의미하며, $M=3$인 경우의 실패 횟수는 프로세서의 수가 3인 경우에 Q값이 1로 결정된 총 횟수를 의미한다.

이 실험에서 생성된 태스크 집합의 평균 태스크 수는 15.473이며 평균 프로세서 이용률은 3.7894이었다. 두 번째 경우에 주기가 증가하지 않는 태스크들의 비율은 $\left\lfloor \frac{n}{2} \right\rfloor$ 으로 결정하였으며, 그들은 주어진 집합 내에서 무작위로 선택되었다.

표 4에서 경우 1은 주기가 증가되는 태스크들이 없는 경우이고, 경우 2는 주기가 증가되는 태스크들이 전체 태스크의 수를 n이라 할 때 $\left\lfloor \frac{n}{2} \right\rfloor$ 인 경우이며, 경우 3은 모든 태스크들의 주기가 증가되는 경우이다. 표 4를 보면 주기가 증가되는 태스크들이 많을수록 최적의 퀀텀 크기가 커지고 스케줄링 실패 횟수도 줄어들고 있음을 볼 수 있다. 또한 태스크 집합 내에서 주기가 증가하는 태스크의 수가 같다 하더라도 개별 태스크의 특성에 따라 결과는 다소 달라질 수 있다. 즉, 주기가 큰 태스크의 주기가 증가하는 경우에는 통상 최적의 퀀텀 크기가 증가하게 되며, 반대의 경우에는 상대적으로 감소하게 된다.

[표 4] 태스크 집합에 대한 실험 결과

항목 / 방법	평균 퀀텀 크기	M=4인 경우의 실패 횟수	M=3인 경우의 실패 횟수
경우 1	5.364	0	89356
경우 2	6.471	0	61247
경우 3	11.258	0	48133

5. 결론

임의의 태스크 집합에 대해서 스케줄링 가능한 최대의 퀀텀 크기는 스케줄링 횟수를 감소시킬 수 있는 최적의 퀀텀 크기이다. 본 논문에서는 다중프로세서 시스템의 실시간 스케줄링 알고리즘에서 최적 퀀텀 크기를 동적으로 결정하기 위한 개선된 방법을 제안하였다.

이전 방법에서는 mode change되는 태스크 집합에 속한 각 태스크의 주기가 감소되는 경우만을 고려하여 최적의 퀀텀 크기를 결정하였는데 본 논문에서는 주기가 감소되는 경우뿐만 아니라, 주기가 증가되는 경우까지도 고려하여 태스크의 도달 함수를 정의하였다. 또한 도달점을 구하기 위한 다항식 함수를 제안하고 그 결과를 증명하였으며, 최적의 퀀텀 크기를 동적으로 결정하는 방법을 제안하였다.

제안된 방법은 기존의 방법에 비해 평균 퀀텀 크기가 현저히 증가하였고 실패 횟수는 상당히 줄어들어 그 효율성을 확인할 수 있었다.

본 논문에서 제안된 방법은 향후 퀀텀 크기 증가에 따른 스케줄러 동작 횟수의 감소나 CPU 클럭 감소에 따른 퀀텀 크기의 결정 등을 통해서 저전력 설계에도 응용될 수 있을 것이다.

참고문헌

[1] 김인국, *흐름 공정 모델의 효율적인 실시간 스케줄링*, 아주대학교 박사학위 논문, 1995.

[2] 김인국 외, “Mode Change 환경에 적합한 동적 퀀텀 크기 스케줄링”, 콘텐츠학회논문집, 제6권, 제9호, pp.28-41, 2006.

[3] C. L. Liu and J. W. Layland, "Scheduling Algorithm for Multiprogramming in a hard real-time environment," JACM, Vol.20. pp.46-61, 1973.

[4] D. Zhu, D. Mosse, and R. Melhem, "Multiple-Resource Periodic Scheduling Problem: how much fairness is necessary?," Real-Time Systems Symposium, Proceedings of the 24th IEEE Real-time Systems Symposium, pp.142-151, Dec. 2003.

[5] J. Anderson and A. Srinivasan, A New Look at Pfair priorities, Technical report, Dept of Computer Science, Univ. of North Carolina, 1999.

[6] J. Anderson and A. Srinivasan, "Early-release fair scheduling," Proceedings of the 12th Euromicro Conference on Real-time Systems, pp.35-43, June. 2000.

[7] J. Anderson and A. Srinivasan, "Pfair Scheduling: Beyond Periodic Task Systems," Proceedings of the 7th International Conference on Real-Time Computing Systems nad Applications, pp.297-306, Dec. 2000.

[8] J. Anderson, A. Block, and A. Srinivasan,

"Quick-release Fair Scheduling," Proceedings of the 24th IEEE Real-time Systems Symposium, pp.130-141, Dec. 2003.

[9] S. Baruah, J. Gehrke, and C. G. Plaxton. "Fast Scheduling of Periodic Tasks on Multiple Resource," Proceedings of the 9th International Parallel Processing Symposium, pp.280-288, Apr. 1995.

[10] S. Baruah, N. Cohen, C. G. Plaxton, and D. Varvel, "Proportionate Progress: A notion of fairness in resource allocation," Algorithmica, Vol.15, pp.600-625, 1996.

김 남 진(Nam-Jin Kim) **[정회원]**

- 1996년 6월 : 단국대학교 물리학과 (이학사)
- 2000년 2월 : 단국대학교 전자계산학과 (이학석사)
- 2003년 2월 : 단국대학교 전자계산학과(박사수료)

<관심분야>
운영체제, 실시간시스템, 임베디드 시스템

김 인 국(In-Guk Kim) **[정회원]**

- 1982년 2월 : 단국대학교(학사)
- 1985년 5월 : 미국 에모리대학교 (석사)
- 1995년 8월 : 아주대학교(박사)
- 1986년 3월 ~ 현재 : 단국대학교 컴퓨터학부 교수
- 2001년 9월 ~ 2003년 5월 : 미국 뉴멕시코공과대학 방문교수

<관심분야>
운영체제, 실시간시스템, 임베디드 시스템