Ядро операционной системы, драйвер КАМАК

Технология построения систем реального времени, к которым относится и описываемая система сбора и управления, опирается на развитую систему планирования и запуска параллельных процессов (time sharing). Реализация этой технологии на бездисковых микро-ЭВМ (Ерухимов и др., 1989) требовала в свое время практически полной авторской разработки микро-ядра, загружаемого вместе с программой регистрации. В настоящее время развитая операционная система легко помещается в память персонального компьютера и выполняет основную долю нагрузки по диспетчеризации процессов. Поэтому достаточно было дополнить стандартное ядро операционной системы драйвером специализированной аппаратуры, а сервисные программы, работающие с аппаратурой через этот драйвер, запускать как независимые задачи.

Структура драйвера КАМАК построена в соответствии с требованиями системы UNIX, обязывающей работу пользовательских программ с внешними устройствами как с файлами. Поэтому драйвер содержит точки входа, необходимые для выполнения следующих функций: open(), close(), read(), ioctl(). С помощью параметров, передаваемых через эти системные функции можно соответственно открывать и закрывать устройство, читать данные из буферной области памяти, а также управлять любым из модулей КАМАК в рамках описанных операций. Последняя функция драйвера позволяет также опрашивать или менять содержимое флажков и областей передачи параметров между пользовательской задачей и ядром. Например, включение таких режимов, как синхронизация системных часов, калибровка, перевод в режим записи данных, пуск каретки и т.п. производится с помощью соответствующей функции ioctl(). Список основных ioctl() драйвера приведен в табл. 3.

Для получения практически независимой работы нескольких параллельных процессов со своими модулями КАМАК тело каждой из перечисленных функций содержит до восьми точек входа, приписанных к соответствующему псевдоустройству: camac, camac1, camac2,..., camac7. Так, например, camac1 служит для чтения флагов состояния, а camac5 -- для управления кареткой. Все псевдоустройста, кроме предназначенных только для чтения флагов, самоблокируются для монопольного использования при открытии и освобождаются функцией close(). Это предотвращает задание противоречивых команд для внешних устройств.

Кроме модулей обслуживания системных запросов, драйвер содержит подпрограмму обработки прерываний от импульсов одного из модулей КАМАК, который совместно с программным счетчиком образует программно-аппаратные системные часы ``звездного'' времени. Все синхронные процедуры располагаются в этой подпрограмме. Последовательность синхронных процедур изображена на рис.6.

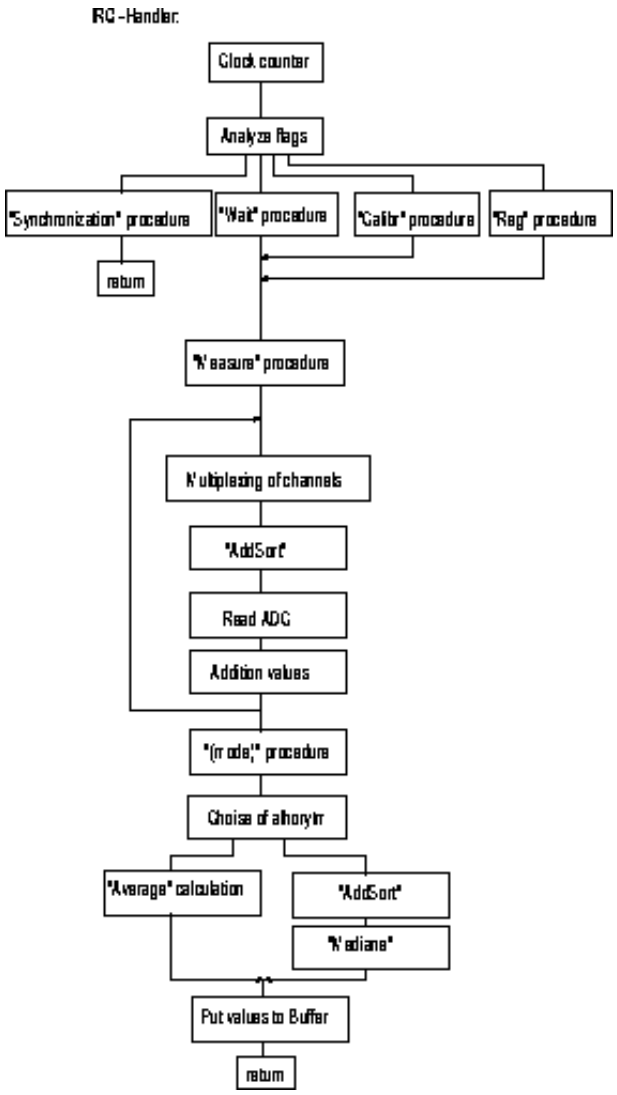

Figure 6: Подпрограмма обработки прерываний от таймера KAMAK.

Как видно из рисунка, подпрограмма содержит несколько ветвей, обуславливающих четыре основных режима работы:

1. Синхронизация системных часов ``звездного'' времени.
2. Ожидание изменения флагов.
3. Калибровка радиометров.
4. Регистрация данных.

Из всех этих режимов только первый запрещает опрос АЦП и вызов процедур управления кареткой, поскольку требует максимальной ``реактивности'' на воздействие импульсов синхронизации часов. В остальных режимах сначала производится чтение данных датчика положения каретки -- Measure(), затем в цикле опрашиваются все необходимые каналы радиометров. Во время относительно долгой процедуры преобразования данных в АЦП вызывается аддитивная сортировка (Кнут, 1978) ранее считанных данных внутри заданного ``окна'' осреднения. То есть эта процедура оптимизирована таким образом, что не требует дополнительных затрат процессорного времени. Кроме сортировки после опроса АЦП производится линейное осреднение данных каждого канала. Тем самым в процедуре считывания предусмотрена возможность получения ``робастных'' (Ерухимов и др., 1988) оценок регистрируемых данных.

После опроса каналов вызывается одна из динамических процедур управления приводом каретки с радиометрами. Адрес процедуры записывается в указатель -- (*mode)() с помощью

системных вызовов ioctl().

Динамические процедуры аналогичны разработанным для системы управления кареткой (Черненков, Ерухимов, 1990) и могут быть как очень простыми типа return, так и довольно сложными, например, отработка ошибки позиционирования. Однако непременным условием такой процедуры является минимальное время исполнения с завершеннием до следующего цикла прерывания от таймера.

Обработка прерывания от таймера заканчивается процедурами анализа завершения накопления данных, и при необходимости, помещением результата осреднения в буферную область памяти. Доступ к этим данным для записи и визуализации осуществляется через системные вызовы read() и ioctl(). При этом используются специальные указатели на начало области данных и их количество.

Vladimir Chernenkov
Wed Nov 27 15:47:58 MSK 1996