Опыт использования процесс-ориентированного подхода
при автоматизации на платформе Adruino

Розов А. С.
Институт автоматики и электрометрии СО РАН

Введение

Появление дешевых серийных микроконтроллеров обеспечило возможность применения технологий автоматизации в задачах, для которых комлексная автоматизация с использованием дорогостоящих ПЛК оказывается нецелесообразной.

Применение готовых микроконтроллерных платформ массового производства устраняет необходимость разработки сложной аппаратной составляющей и снижает требования к квалификации разработчиков. Это позволяет значительно снизить временные и финанасовые затраты на разработку и тиражирование систем в задачах так называемой малой автоматизации.

Популярный стандарт в этой области – Arduino-совместимые платформы (Рис. 1). Arduino предоставляет открытую аппаратную архитектуру в совокупности со средой разработки на языке Си и набором высокоуровневых библиотек для работы с внешним устройствами.

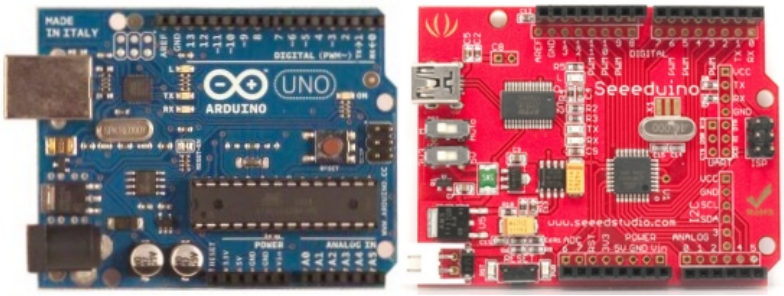

Рис. 1. Arduino (слева) и Seeeduino v2.21 – пример Arduino-совместимой архитектуры

На данный момент существует два метода разработки под Arduino – с использованием встроенных функций среды Arduino, и путем непосредственной работы с регистрами микроконтроллера. В обоих случаях разработка ведется на языке Си. При этом использование встроенных функций среды Arduino не позволяет в полной мере утилизировать возможности микроконтроллера, а непосредственная работа с регистрами требует относительно высокой квалификации разработчика.

Кроме того, управляющие алгоритмы обладают рядом характерных особенностей, таких как событийность, цикличность, необходимость синхронизации с внешними процессами и логический параллелизм [1].

Эти особенности могут быть учтены при создании алгоритмов управления за счет использования концепции процесса [2] – расширенной модели конечного автомата.

Использование концепции процесса при реализации алгоритма управления на базе Arduino-совместимой архитектуры (Seeeduino) с использованием языка Си подтвердило перспективность подхода. Выявленный недостаток кодирования на Си – отсутствие поддержки концепции на уровне языка, что обуславливает большое количество рутинных операций и высокую вероятность семантической ошибки при кодировании.

Поэтому при создании управляющих алгоритмов целесообразно использовать специализированный язык программирования Reflex [3], ориентированный на решение задач автоматизации. Язык имеет Си-подобный синтаксис и реализует концепцию процесс-ориентированного программирования, в которой программы описываются как совокупность взаимодействующих процессов.

Существенный недостаток языка при применении на платформе Arduino – отсутствие поддержки работы с прерываниями.

Таким образом, на настоящий момент отсутствуют языковые средства, позволяющие эффективно создавать управляющие алгоритмы для Arduino-совместимых архитектур. Одно из возможных направлений создания такого языка – создание расширения языка Рефлекс, ориентированного на задачи малой автоматизации. В работе обсуждаются требования, предъявляемые к языку, а также расширение модели гиперпроцесса для микроконтроллерных платформ, во многом определяющее синтаксис языка.

Требования

В ходе проведенного исследования были сформулированы требования к разрабатываемому языку, учитывающие особенности разработки управляющих алгоритмов для программируемых микроконтроллеров:

- Поддержка синтаксиса языка Си
- Событийность
- Синхронизм
- Логический параллелизм
- Статическая проверка достижимости состояний
- Поддержка разбиения программы на модули
- Поддержка работы с прерываниями

Для максимизации функциональности предлагается полностью сохранить поддержку синтаксиса языка Си, расширив его конструкциями нового языка. Это требование достигается за счет промежуточной трансляции языка в Си-код. Требования событийности и синхронизации обусловлены спецификой управляющих алгоритмов и задач автоматизации. Разбиение программы на независимо компилируемые модули снижает время сборки и существенно упрощает разработку.

Отсутствие такой возможности – один из основных недостатков существующих версий языка Reflex. Эффективное применение языка на микроконтроллерных платформах требует поддержки работы с прерываниями на уровне конструкций языка.

Модель гиперпроцесса

В работе [2] обсуждалась модификация классической модели конечного автомата, лежащая в основе языка Reflex и направленная на применение в алгоритмах управления – процессы и гиперпроцессы.

В формальном виде процесс pi задается четверкой элементов, которые отражают статическую и динамическую информацию о процессе:

$$p_i = (F_i, f_i^1, f_i^{cur}, T_i^p), \text{ где}$$

F_i – множество функций-состояний процесса;

f_i^1 – начальная функция-состояние, $f_i^1 \in F_i$;

f_i^{cur} – текущая функция-состояние, $f_i^{cur} \in F_i$;

T_i^p – текущая продолжительность нахождения процесса в текущей функции-состоянии.

j-я функция-состояние произвольного *i*-го процесса задается парой элементов:

$$f_i^j = (X_i^j, Y_i^j), \text{ где}$$

$X_i^j = \{x_i^{j1}, \ldots, x_i^{jL}\}$ – множество событий;

$Y_i^j = \{y_i^{j1}, \ldots, y_i^{jL}\}$ – множество реакций.

Множество реакций Y_i^j можно разделить на два подмножества: вычислительные реакции Y_i^{jcalc} и Y_i^{jctrl} – реакции, меняющие поток управления путем задания следующей функции-состояния процесса.

$$Y_i^j = Y_i^{jcalc} \cup Y_i^{jctrl}.$$

Среди функций-состояний процесса F_i различаются взаимно непересекающиеся множества *активных и пассивных функций-состояний*, F_i^a и F_i^p соответственно. Функция-состояние процесса пассивна, если его множество реакций пусто. Пассивные функции состояния это f^{NS} «нормальный останов» и f^{ES} «останов по ошибке».

Гиперпроцесс H представляет собой упорядоченный набор процессов, циклически активизируемых с периодом активизации T_H:

$$H = (T_H, P, p_1), \text{ где}$$

T_H – период активизации гиперпроцесса;

P – множество процессов;

p_1 – начальный процесс, $p_1 \in P$.

Процессы исполняются последовательно в порядке описания, и каждый процесс фактически является отдельным потоком управления. Таким образом, модель гиперпроцесса реализует принципы логического параллелизма и событийности.

Приведенная математическая модель разрабатывалась для решения широкого класса задач и удовлетворяет не всем выдвинутым в работе требованиям. Учет специфики разработки систем на базе микроконтроллеров требует модификации модели гиперпроцесса и введения ряда уточнений.

Модификация модели гиперпроцесса для микроконтроллеров

Одна из основных особенностей работы с программируемыми микроконтроллерами – наличие системы прерываний. Для описания алгоритмов управления на микроконтроллерах предлагается модифицировать модель гиперпроцесса, добавив в нее поддержку работы с прерываниями.

Каждую функцию-обработчик прерывания (ISR) предлагается описывать как отдельный гиперпроцесс. Таким образом, в программе одновременно существуют один основной (фоновый) гиперпроцесс, и множество гиперпроцессов обработки прерываний. На практике гиперпроцессы обработки прерываний скорее всего будут содержать значительно меньшее количество процессов чем фоновый гиперпроцесс.

С введением прерываний изменяется также структура множества реакций процесса. К множествам $Y_i^{j\text{calc}}$ и $Y_i^{j\text{ctrl}}$ добавляется также множество реакций $Y_i^{j\text{int}}$, управляющих работой прерываний и задающих взаимодействие между гиперпроцессами:

$$Y_i^j = Y_i^{j\text{calc}} \cup Y_i^{j\text{ctrl}} \cup Y_i^{j\text{int}}.$$

Множество $Y_i^{j\text{int}}$ может включать в себя следующие виды реакций:

- изменение регистров, управляющих прерываниями;
- изменение регистров, непосредственно запрещающих и разрешающих прерывания;
- изменение значений «разделяемых» переменных, используемых для взаимодействия между гиперпроцессами.

Адреса функций-обработчиков в векторах прерываний предлагается определять статически на этапе компиляции. При этом динамическая модификация векторов не является допустимой реакцией и не входит в $Y_i^{j\text{int}}$.

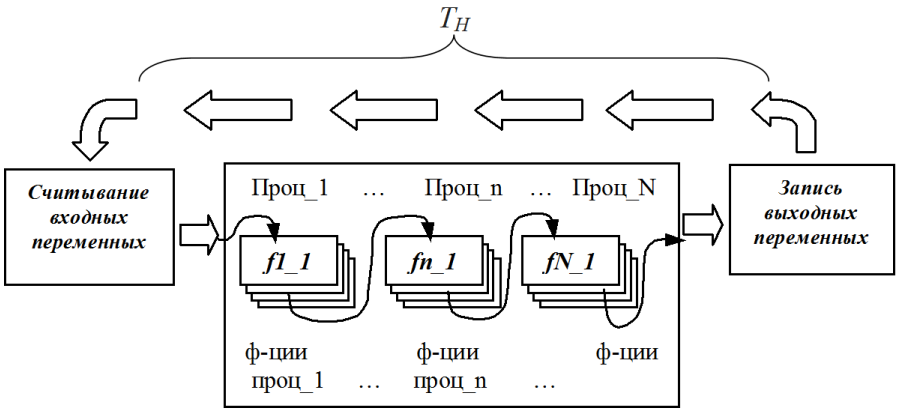

Рис. 2. Циклическое функционирование гиперпроцесса

В описанной в работе [2] модели гиперпроцесс функционирует циклически с неизменным периодом активизации T_H (Рис. 2) При этом для обеспечения необходимой скорости реакции вводится предположение о том, что период активизации гиперпроцесса значительно меньше времени протекания внешних процессов:

$$T_H << T_{\text{event}}.$$

Период активизации определяет время реакции системы на события. Если внутри такта гиперпроцесса может возникнуть прерывание, период активизации может случайным образом возрастать, и его постоянность нарушается. Процедура обработки прерывания в свою очередь также описывается гиперпроцессом с заданным периодом активизации, и ее исполнение также может быть приостановлено другим прерыванием. Таким образом, появление прерываний требует усиления вышеописанного требования следующим утверждением:

$$T_H^{main} + \sum_{k \in I} T_H^k < T_H, \text{ где}$$

T_H^{main} – время исполнения фонового гиперпроцесса;

I – множество всех активных гиперпроцессов обработки прерываний;

T_H^k – время исполнения *k*-ого гиперпроцесса обработки прерывания;

T_H – период активизации фонового гиперпроцесса.

Заключение

Эффективное использование процесс-ориентированного подхода на базе микроконтроллерных платформ требует поддержки работы с прерываниями на уровне языка. Такой язык может быть получен путем расширения языка Reflex с использованием приведенной модификации модели гиперпроцесса. В продолжение работы планируется определить синтаксис языка и реализовать транслятор в язык Си.

Список литературы

1. Зюбин В.Е. Программирование ПЛК: языки МЭК 61131-3 и возможные альтернативы // Промышленные АСУ и контроллеры. 2005. № 11. С. 31-35.
2. Зюбин В.Е. Процесс-ориентировнное программирование. Учеб. пособие / Новосибирск. Новосиб. гос. ун-т. 2011. 194 с.
3. Зюбин В.Е. "Си с процессами" – язык программирования логических контроллеров // Мехатроника, автоматизация, управление. 2006. № 12. С. 31-35.