Постановка задачи .

Необходимо создать программное обеспечение на языке Ассемблера, реализующее ПИ-регулятор. Программа должна быть реализована, используя целочисленные значения. При написании программы была использована среда разработки и отладки программного кода Atmel Studio.

Теоретическое введение.

ПИД - регулятором называется устройство, применяемое в контурах управления, оснащенных звеном обратной связи. Относится к наиболее распространенному типу регуляторов. В нем используется пропорционально-интегрально-дифференциальный закон регулирования. Данные регуляторы используют для формирования сигнала управления в автоматических системах, где необходимо достичь высоких требований к качеству и точности переходных процессов.

Пропорциональная составляющая вырабатывает выходной сигнал, противодействующий отклонению регулируемой величины от заданного значения, наблюдаемому в данный момент времени. Он тем больше, чем больше это отклонение.

Интегрирующая составляющая пропорциональна интегралу по времени от отклонения регулируемой величины. Её используют для устранения статической ошибки. Она позволяет регулятору со временем учесть статическую ошибку.

Дифференцирующая составляющая пропорциональна темпу изменения отклонения регулируемой величины и предназначена для противодействия отклонениям от целевого значения, которые прогнозируются в будущем. Отклонения могут быть вызваны внешними возмущениями или запаздыванием воздействия регулятора на систему.

Управляющий сигнал ПИД-регулятора получается в результате сложения трех составляющих: первая пропорциональна величине сигнала рассогласования, вторая — интегралу сигнала рассогласования, третья — его производной. Его назначение состоит в поддержании заданного значения x с помощью изменения величины $u(t)$.

Схема, иллюстрирующая принцип работы ПИД-регулятора на рисунке 1.

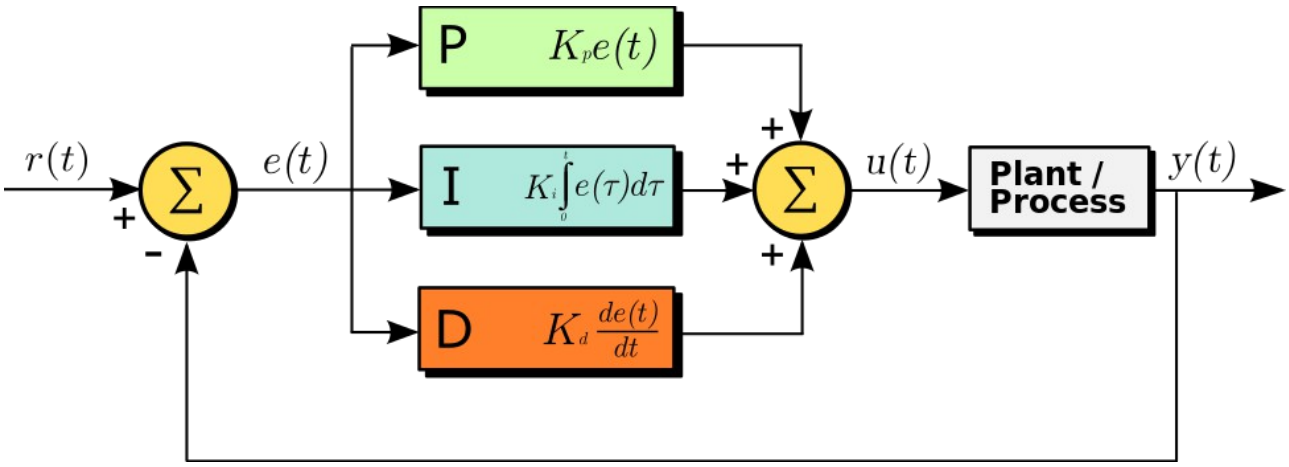


Рисунок 1. принцип работы ПИД-регулятора.

Соответствующая формула ПИД-регулятора для воздействия *u(t)* определяется следующим образом:

$$u(t) = P + I + D = K_p\, e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de}{dt}$$

Где *e(t)* — текущая ошибка, *Kp, Ki, Kd* — коэффициенты.

В этой формуле есть все три коэффициента пропорциональности для каждого из трех слагаемых. В вычислительной технике нет возможности реализовать такую формулу, поэтому в программной реализации имеет место использование реккурентной формулы:

В итоге, задача была разделена на несколько простых частей, каждая из которых считала отдельные значения каждого из звеньев и затем суммировала все три звена.

Пропорциональное звено.

Пропорциональная составляющая вырабатывает выходной сигнал, противодействующий отклонению регулируемой величины от заданного значения, наблюдаемому в данный момент времени. Он тем больше, чем больше это отклонение. Если входной сигнал равен заданному значению, то выходной равен нулю.

Однако при использовании только пропорционального регулятора значение регулируемой величины никогда не стабилизируется на заданном значении. Существует так называемая статическая ошибка, которая равна такому отклонению регулируемой величины, которое обеспечивает выходной сигнал, стабилизирующий выходную величину именно на этом значении.

Чем больше коэффициент пропорциональности между входным и выходным сигналом (коэффициент усиления), тем меньше статическая ошибка, однако при слишком большом коэффициенте усиления при наличии задержек (запаздывания) в системе могут начаться автоколебания, а при дальнейшем увеличении коэффициента система может потерять устойчивость.

На рисунке 2 показано влияние пропорционального коэффициента на выходной сигнал.

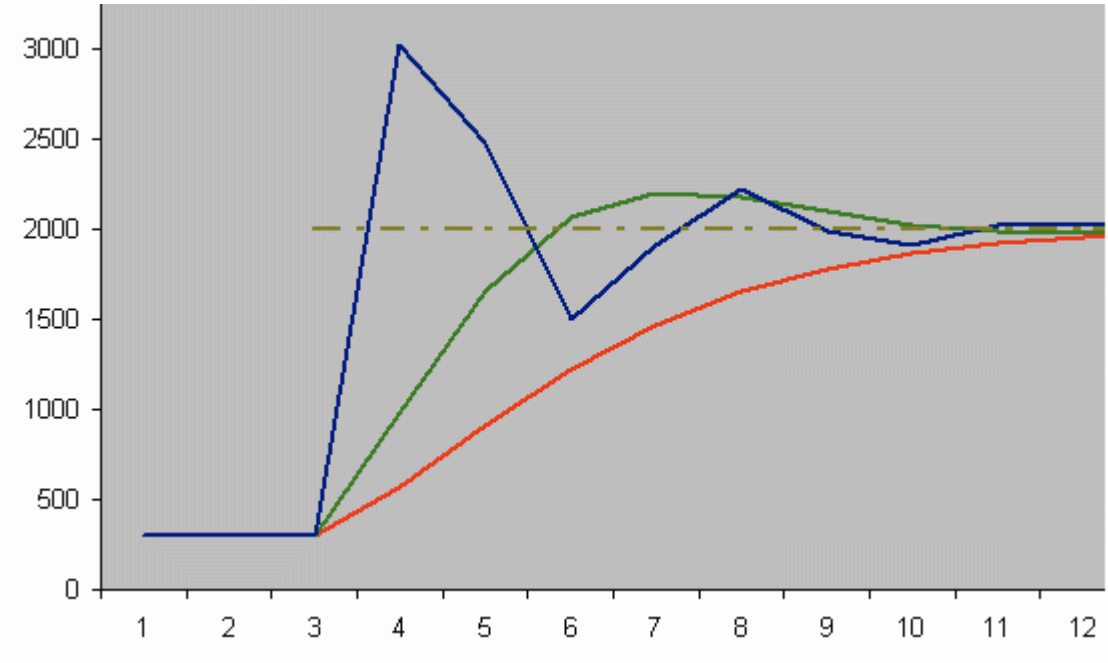


Рисунок 2. Влияние пропорционального коэффициента на сигнал.

Интегральное звено.

Интегрирующая составляющая пропорциональна интегралу по времени от отклонения регулируемой величины. Её используют для устранения статической ошибки. Она позволяет регулятору со временем учесть статическую ошибку. Как и любой интегратор, она накапливает ошибку, тем самым позволяя сгладить резкий эффект от пропорциональной составляющей.

Если система не испытывает внешних возмущений, то через некоторое время регулируемая величина стабилизируется на заданном значении, сигнал пропорциональной составляющей будет равен нулю, а выходной сигнал будет полностью обеспечиваться интегрирующей составляющей. Тем не менее, интегрирующая составляющая также может приводить к автоколебаниям при неправильном выборе её коэффициента.

На рисунке 3 изображено влияние интегральной состовляющей на выходной сигнал.

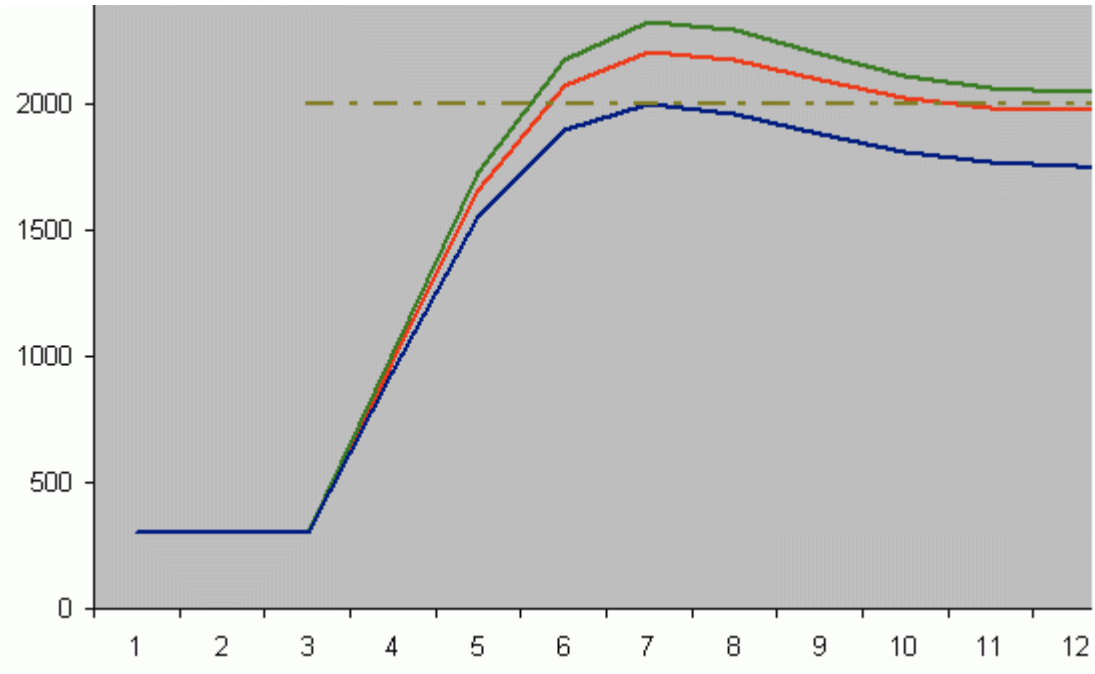


Рисунок 3. Влияние интегрального коэффициента

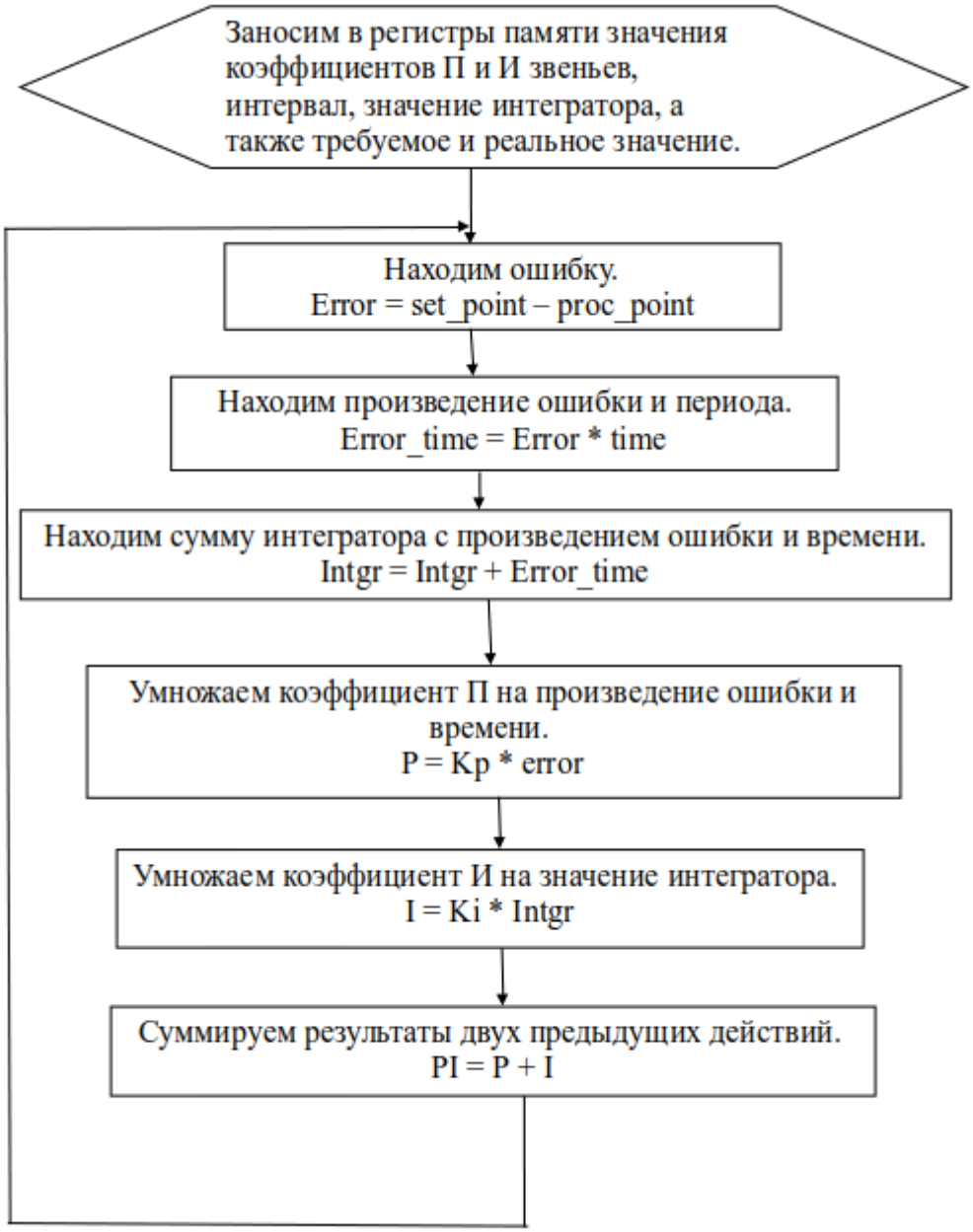
Реализация

С целью отработки формулы ПИ-регулятора была разработана программа на языке С.

Для выходного сигнала регулятора необходимо рассчитать два звена, а затем их суммировать. Таким образом, их рассчет был реализован следующим образом:

```
error = set_point — proc_point;
error_time = error * time;
Intgr = Intgr + error_time
P = Kp * error
I = Ki * I
PI = P + I
```

Блок-схема программы приведена ниже.



Методика и результаты тестирования.

С целью проверки работоспособности программы, есть необходимости проверки его в среде разработки Atmel studio. Для этого зададим соответствующие переменные, а затем пошагово пройдем по коду для проверки правильности исполнения каждой арифметической операции и корректности построения программы в целом. Выходной сигнал равен сумме пропорциональной и интегральной составляющих.

Первым делом необходимо инициализировать переменные.

```
ldi set_, 7
ldi proc, 5
ldi Kp, 3
ldi Ki, 10
ldi time, 1
ldi Intgr, 0
```

Рисунок 4. Инициализация переменных

Далее, пошагово пройдем строчки кода и проверим в специальном окошке Watch значения этих переменных.

set_	7
proc	5
Kp	3
Ki	10
time	1
Intgr	0

Рисунок 5. Проверка значений после инициализации

После этого переходим к самому блоку кода, где происходит реализация ПИ-регулятора. Первое действие это рассчет ошибки регулирования. Чтобы получить ее значение, первое что нужно сделать, это перенести заданное значение в регистр ошибки и затем необходимо отнять от этого значения — значение фактическое. Таким образом, у нас получается, что значение регистра error сначала становится равным 7, а затем, после операции вычитания, становится равным 2.

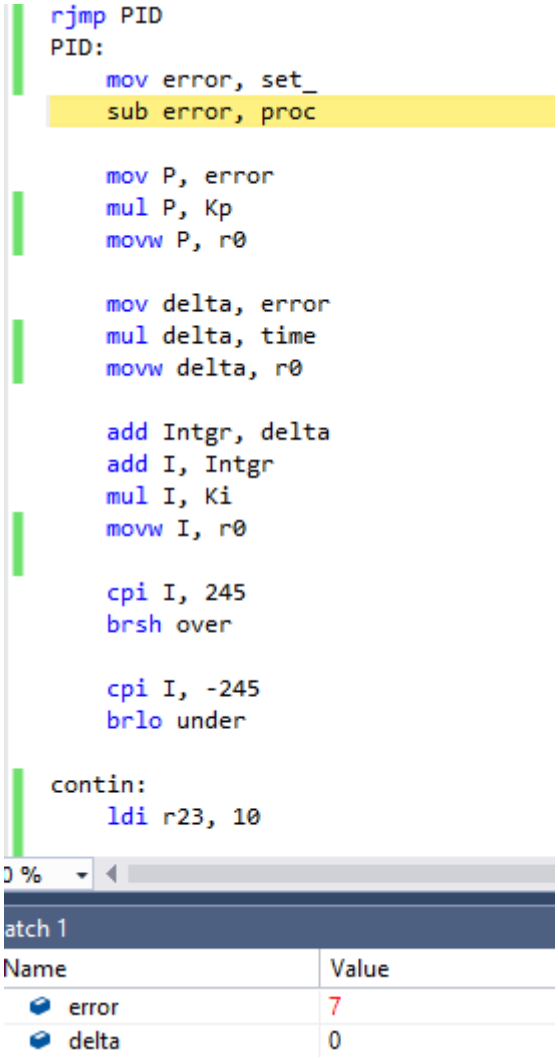


Рисунок 6. Значние ошибки после занесения заданного значения

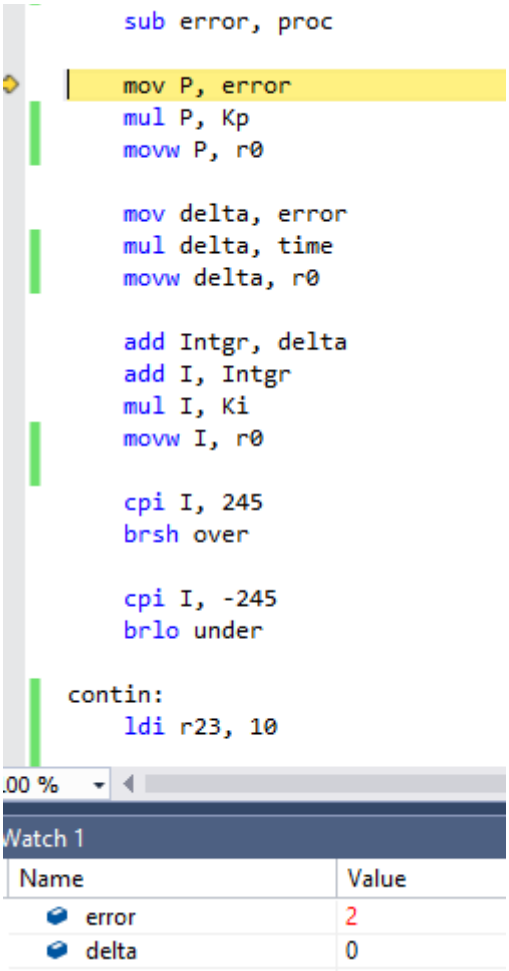


Рисунок 8. Значение ошибки, после вычитания из нее фактического значения

Далее, чтобы реализовать П-звено, нужно перемножить полученную ошибку на коэффициент пропорциональности. Для этого заносим значение ошибки в регистр P, умножаем P на коэфиициент Kp и копируем в результирующий регистр значение из исходного регистра.

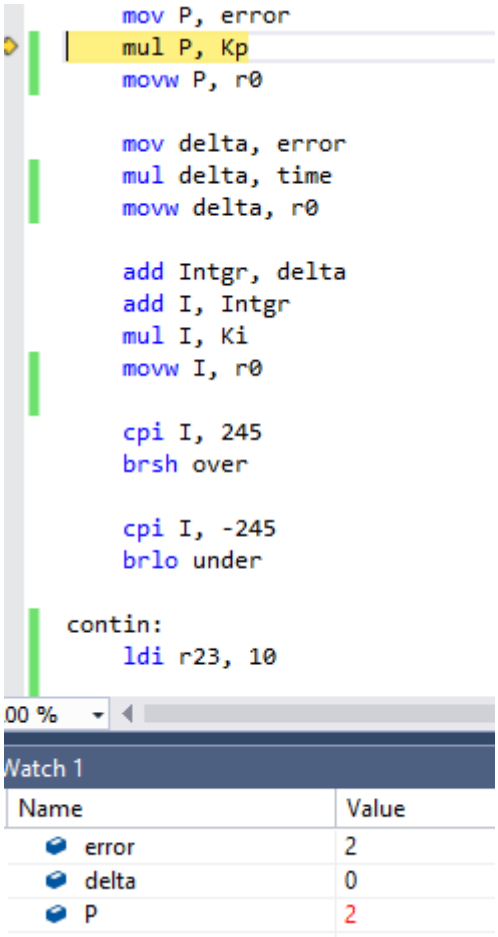


Рисунок 9. Занесение значения ошибки в регистр P

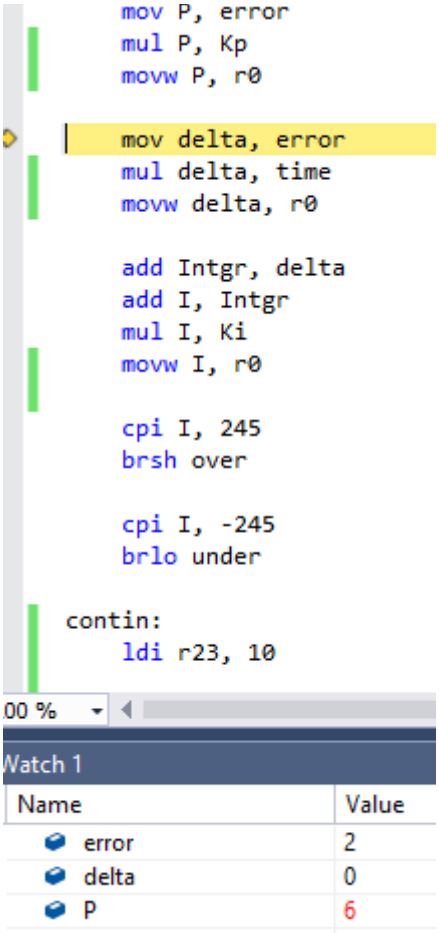


Рисунок 10. Результат после умножения на коэффициент Kp

После этого идет реализация И-звена. Он необходим, чтобы устранить ошибки установившегося состояния, путем интегрирования ошибки в течение периода времени. Из-за нелинейности сигнала в реальных условиях, есть необходимости задать диапозон для преодоления интегральных условий.

Для начала перемножим ошибку и время.

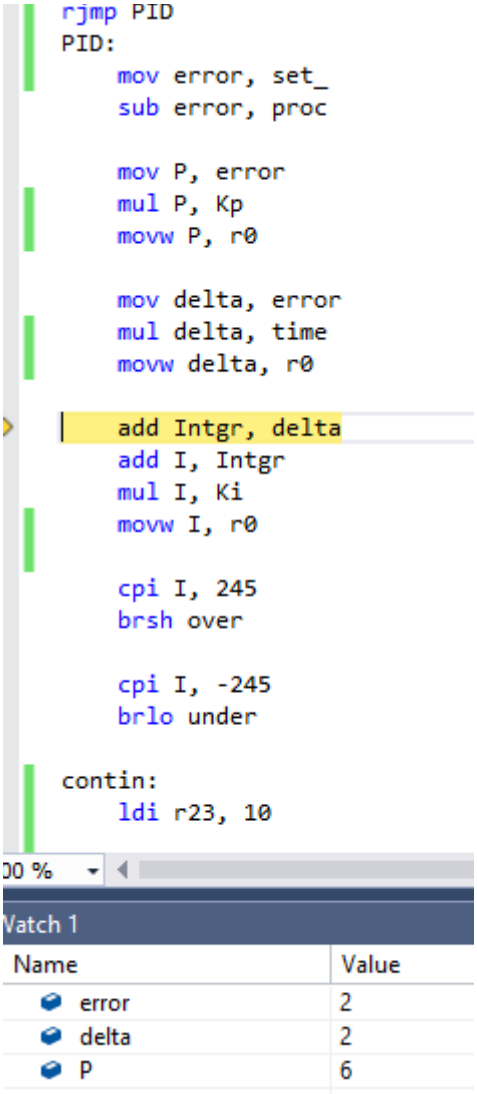


Рисунок 11. Delta – результат перемножения ошибки и времени

Затем идет рассчет интегратора — добавляем к интегратору значение из регистра delta. После добавить значение интегратора к И-звено и перемножить звено на соответствующий коэффициент.

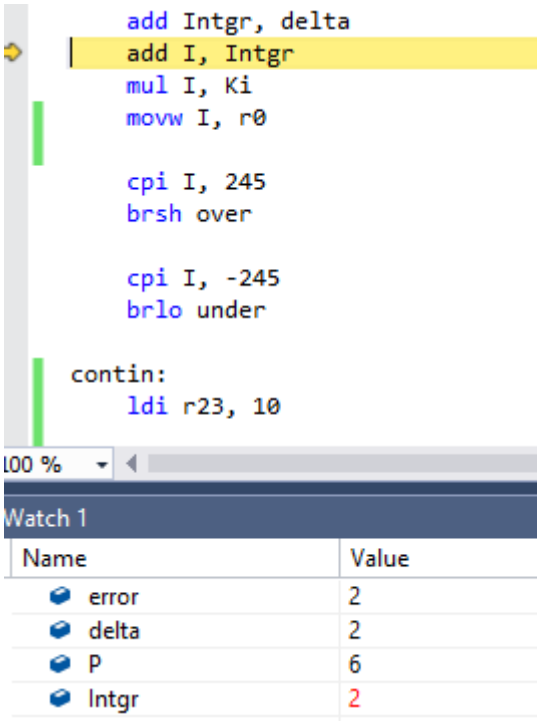


Рисунок 12. Занесение delta в интегратор

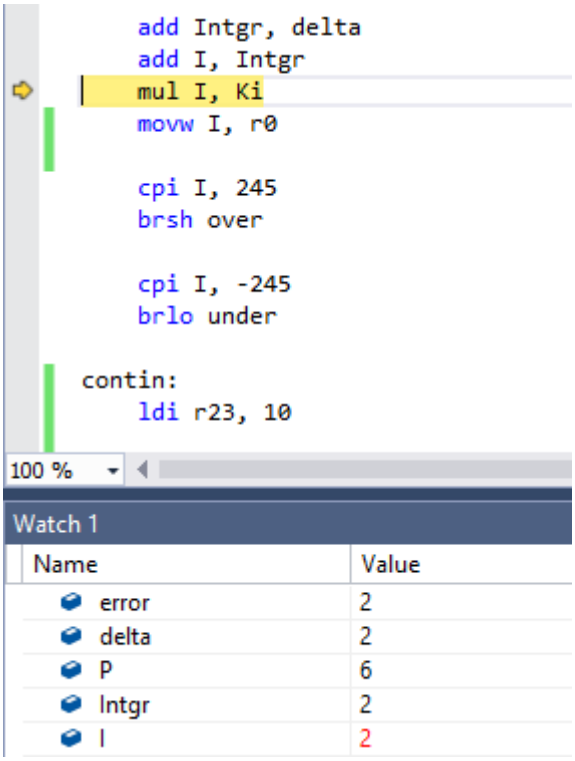


Рисунок 13. Суммирование интегратора и И-звена

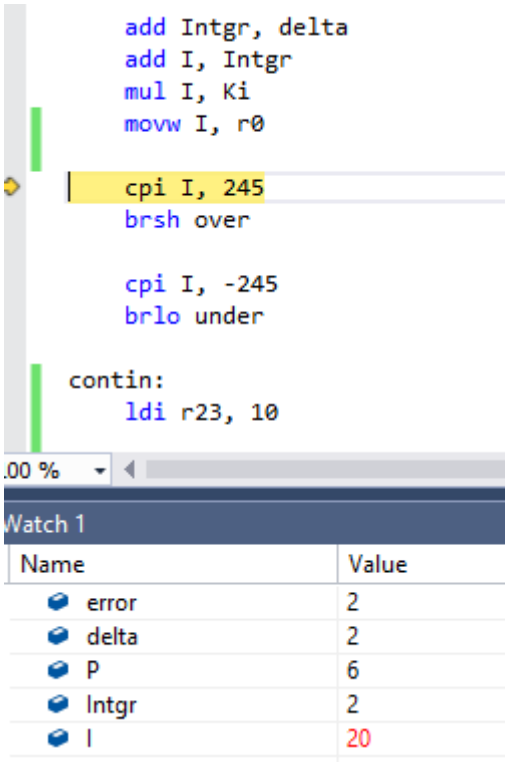


Рисунок 14. Перемножение И-звена на коэффициент

При реализации интегрального звена следует ограничить рост интегральной суммы. Это можно реализовать с помощью операции сравнения с константой и затем, если результат больше заданного, то присвоить максимальное значение, если меньше, то минимальное.

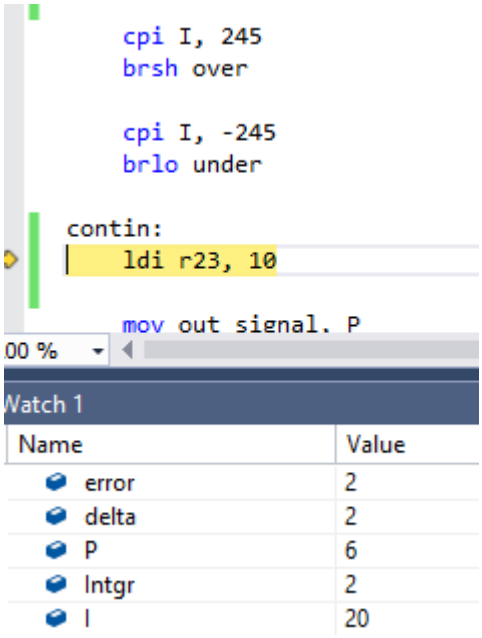


Рисунок 15. Проверка на выход за границы

При данной итерации значение попадает в заданный промежуток. Поэтому переход к рассчету выходного сигнала. Для этого, заносим значени П-звена в регистр out_signal, а затем добавляем к этому регистру результат И-звена и тем самым реалуется ПИ-регулятор.

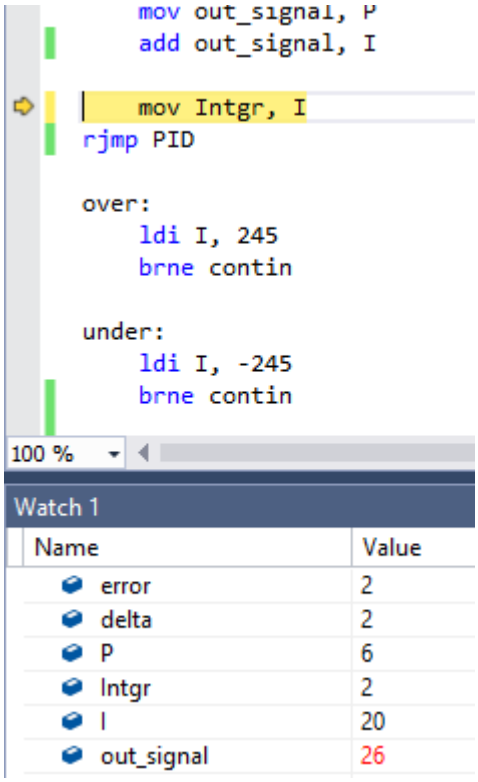


Рисунок 16. Реализация ПИ-регулятора

После этого, программа делает задержку и снова переходит к рассчету всех значений и работает в бесконечном цикле с прерываниями. Подводя итоги, можно сказать, что алгоритм был реализован. Однако при симуляции процесса недостаточно выявить плюсы и минусы конкретной программы. Для этого необходимо проводить тестирование на реальных объектах.

ПИ-регулятор является частным случаем ПИД-регулятора, при это является наиболее распространенным благодаря своим достоинствам, например способность обеспечить нулевую статическую ошибку регулирования и простоты настройки, так как настраивается только два коэффициента.

Листинг кода.

```
.def error = R17
.def set_ = R28
.def proc = R19
.def P = R20
.def delta = R29
.def time = R30
.def Intgr = R22
.def I = R23
.def Kp = R24
.def Ki = R25
.def out_signal = R26

rjmp stack
.ORG 0x0003
rjmp PID

ldi set_, 10
ldi proc, 8
ldi Kp, 10
ldi Ki, 30
ldi time, 1
ldi Intgr, 0

stack:
	ldi R16, HIGH(RAMEND)
	out SPH, R16
	ldi R16, LOW(RAMEND)
	out SPL, R16

	ldi R16, 1 << OCIE2
	out TIMSK, R16
	ldi R16, (1 << WGM21) | (1 << CS22) | (1 << CS21) | (1 << CS20)
	out TCCR2, R16
	ldi R18,
	out OCR2, R18
	ldi R16, (1 << PB3)
	out DDRB, R16
	out PORTB, R16
sei

start:
rjmp start

brcc start

PID:
	sbis portb, 3
	mov error, set_
	sub error, proc

	mov P, error
	mul P, Kp

	mov delta, error
	mul delta, time
```

```
        add Intgr, delta
        add I, Intgr
        mul I, Ki

        cpi I, 250
        brsh over

        cpi I, -250
        brlo under
contin:
        mov out_signal, P
        add out_signal, I

        mov Intgr, I
        rjmp PID

over:
        ldi I, 250
        brne contin

under:
        ldi I, -250
        brne contin
reti
```