При роботі з мікроконтролерами сімейства PIC, який був обраний для керування апаратною частиною системи, для написання програм керування існує можливість використання лише двох мов програмування: мови низького рівня асемблера та мови високого рівня С.

Програма розроблена на мові асемблера виконується швидше та потребує менше ресурсів ніж аналогічна програма написана на С. Написання програми для мікроконтролера на асемблері дозволяє краще взаємодіяти із вбудованими апаратними засобами мікроконтролера.

Іще однією превагою використання асемблера на етапі розробки є те, що деякі середовища схемотехнічного моделювання, наприклад Porteus ISIS, дозволяє використовувати не тільки скомпільовані програми, а і вихідний текст програми на асемблері. Це може суттєво прискорити розробку простих рішень та відмовитись від встановлення спеціальних інтегрованих середовищ розробки.

Однак за таку оптимізацію доводиться платити більшими затратами часу на розробку програми. Оскільки для розробки на асемблері необхідно добре розуміти архітектуру конкретного мікроконтролера, для якого ведеться розробка.

Слід також зауважити, що асемблер для мікроконтролерів PIC значно відрізняється від звичного асемблера архітектури x86. Більше того кожна серія мікроконтролерів має власну реалізацію мови асемблера. Проте фірма Microchip оснащує свої продукти детальною документацією в якій описано як апаратні особливості архітектури так і про особливості взаємодії з ними, тому досвідченим розробникам апаратури використання асемблера не завдасть значних труднощів.

Іншим можливим рішенням є використання мови С, переваги якої складно переоцінити. На відміну від асемблера мова С більш зрозуміла для розробники та дозволяє в деякій мірі абстрагуватись від апаратних особливостей архітектури мікроконтролера. Що в свою чергу зробить вихідний код більш гнучким та спростить його перенесення на інший тип мікроконтролера. За роки існування цієї мови в інтернеті накопчелась велика кількість алгоритмів для рішення майже будь-якої задачі. Тому більшість розробників віддають перевагу саме цій мові.

Однак програмування на С потребує використання спеціальних інтегрованих середовищ розробки. На щастя фірма Microchip поставляє таке середовище, що розроблене спеціально для мікроконтролерів PIC.

Таким середовищем є MPLAB фірми Microchip. Він має набір інструментів для розробки та налагодження програмного коду для керування мікроконтролерами, такі як компілятори, лінкувальники, засоби для слідкування за ходом виконання коду. Для кожної серії мікроконтролерів необхідно використовувати різні компілятори, тому MPLAB створений з окремих модулів та дозволяє встановлювати різні інструменти, залежно від ситуації.

Незважаючи на всі переваги та недоліки кожної з мов програмування вирішальне значення мала наявність бібліотеки для роботи з ПК через USB.

Універсальний драйвер Microchip для роботи з USB під керуванням операційної системи Windows (Microchip General Purpose USB Windows), скдається з трьох частин: бібліотек для керування USB-модулем мікроконтролера, драйвер для операційної системи та DLL бібліотека, що полегшує програмну взаємодію з драйвером.

Разом з бібліотекою поставляються вихідні коди програм для мікроконтролерів різних типів, та демонстраційні приклади, з яких можна скомпілювати програми прошивки. Кожен приклад побудований таким чином, щоб можна було видалити файли, що не є частиною бібліотеки і використати його як основу для створення власних програм.

Один з таких демонстраційних проектів і було взято за основу. Він має вже готову архітектуру та налаштовану конфігурацію мікроконтролера. Інші засоби є досить простими і можуть бути з легкістю використані в будь-якому проекті.

Спершу слід визначитись з особливостями роботи готової частини, наприклад тактовим сигналом. Для роботи з USB потрібна висока стабільність тактових імпульсів, тому неможна використовувати звичайну конфігурацію. Для стаібізації тактової частоти до мікрокотролера підключений кварцовий резонатор з часттою 1 МГц. З урахуванням заданої конфігурацію мікроконтролера його тактова частота становить 5 МГц. Це дуже важливий параметр, оскільки від нього залежить частота переключення таймерів, а отже і заначення, що необхідно задати, щоб отримати потрібні інтервали часу.

Таймери та переривання

Керування напругою колектор-емітер відбувається за допомогою широтно-імпульсної модуляції (ШІМ), тобто результуюча напруга залежить від шпаруватості імпульсів що подаються на вхід схеми керування. Це створює необхідність у відносно високій стабільності генерованих імпульсів, яку не можна забезпечити при керуванні з основного циклу програми. В цьому випадку можна скористатись вбудованими таймерами мікроконтролера.

Використання таймерів, як і більшості периферійних модулів засновано на перериваннях. Переривання це механізм, що дозволяє перервати виконання програми та передати керування іншій ділянці коду. По завершенню виконання цієї ділянки керування буде передано туди де було перервано виконання основного коду. Це дозволяє швидко реагувати на настання певних подій.

Мікроконтролери сімейства PIC18 мають дворівневу систему переривань, тобто переривання можуть мати два пріоритети: високий і низький, відповідно і дві функції обробники. [<http://pro-diod.ru/programms/pic-micro/interrupt-v-pic18.html>]

Компілятор мови С надає досить зручний засіб для роботи з обробниками переривань. За документацією, щоб визначити функцію як обробник необхідно додати ключове слово *interrupt*, та за необхідності пріоритет (наприклад *low\_priority*) у визначення функції, яке матиме наступний вигляд: *void interrupt low\_priority  low\_isr (void)*. Це означає що функція буде викликана при появі переривання з низьким пріоритетом.

На жаль в нових версіях компілятора таке визначення обробників переривань не працює. Однак адреси потрібних функцій можна «вручну» занести у відповідні вектори переривань. Вектор — це адреса в програмній пам'яті, за якою відбувається автоматичний перехід при виникненні будь-якого дозволеного переривання.

Як було зазначено даний мікроконтролер має два типи переривань, а отже і два вектора:

* 0x08 — вектор переривань з високим пріоритетом;
* 0x18 — вектор переривань з низьким пріоритетом.

Слід пам’ятати, що вразі відключення пріоритетів то буде використовуватись лише один вектор 0x08.

Щоб налаштувати переривання необхідно дозволити переривання глобально та периферійні встановленням біті GIE та PEIE регістру INTCON відповідно та пріоритети встановленням IPEN регістру RCON.[<http://subscribe.ru/archive/comp.soft.prog.pic/200703/13210529.html>]

програмне керування зміною напруги

В минулому розділі вже було описано схемну реалізацію понижуючого перетворювача постійного струму на основі широтно-імпульсної модуляції. Тут буде наведено програмну реалізацію алгоритму керування ШІМ.

Найкращий варіант роботи керованого перетворювача постійного струму на основі широтно-імпульсної модуляції це використання апаратних засобі ШІМ (PWM, Pulse-Width Modulation) мікроконтролера. Для цього в мікроконтролері передбачено модуль CCP. Абревіатура розшифровується як Capture/Compare/PWM (Захват/Порівняння/ШІМ). Обрана модель мікроконтролера має два незалежних модулі CCP1 та CCP2. Однак вони, є не зовсім незалежними, оскіьки працюють на одній частоті та використовують однаковий спосіб підключення, проте є можливість задавати шпаруватість для кожного з них незалежно.

Для роботи модуля потрібно налаштувати виводи RC2 (CCP1) та RC1 (CCP2) на вихід, та активувати таймер 2. При використанні модулів CCP цей таймер буде рахувати такти та керувати довжиною імпульсів, а отже не доступний для інших функцій.[<http://picdevices.ru/eksperimentyi/pwm-dac-lpf-ili-kak-eto-po-nashemu-shim-tsap-fnch.html?doing_wp_cron=1464891333.4888598918914794921875>]

[<http://catcatcat.d-lan.dp.ua/skachat/primeryi-postroeniya-koda-programm-dlya-pic-kontrollerov/shirotno-impulsnaya-modulyatsiya/>]

Ініціалізація модуля CCP режимі ШІМ включає наступні етапи. Перш за все потрібно встановити режим роботи модуля та встановити відповідні виводи порту С на вихід. За режим роботи відповідають біти 0–3 регістру CCP1CON. Вибір режиму ШІМ відбувається встановленням бітів 2 та 3 (останні два ігноруються і можуть мати будь-яке значення).[<http://portal.tpu.ru/SHARED/p/PEST/pwm.pdf>]

За період імпульсів відповідає регістр PR2, що має 8 розрядів, а отже його значення не може перевищувати 255. Фактичне значення періоду розраховується за наступною формулою:

Ppwm = (PR2 + 1) \* 4 \* Tosc \* Nprescale,

де Ppwm — період імпульсів;

Tosc — тактова частота роботи мікроконтролера;

Nprescale значення переддільника таймера. Цей параметр може приймати три значення 1, 4 або 16. Для цього в біти 0–1 регістра T2CON потрібно встановити в значення 00, 01 або 11 відповідно.

За формулою на основі потрібної частоти імпульсів можна розрахувати значення PR2. Краще спершу взяти значення переддільника рівним 16 та розрахувати PR2, якщо значення перевищить 255 то взяти менше значення переддільника та повторити розрахунок. Використання регістра PR2 та таймера можна побачити на рисунку

Рисунок

Видно, що зміна значення напруги з високоги на низькие і навпаки відбувається коли коли таймер досягне деякого значення. Це означає, що частота імпульсів кратна частоті переключень таймера, який в свою чергу залежить від тактової частоти мікроконтролера. Очевидно, що підвищення частоти можна досягти зменшенням граничного значення лічильника. Однак таке зменшення призведе до того, що імпульс триватиме меншу кількість тактів і поділити його можна буде на меншу кількість частин. Цей параметр називається розрядністю ШІМ. Від дорівнює кількості біт, в яку можна записати число відношення частоти роботи мікроконтролера до необхідної частоти імпульсів. Наприклад, якщо тактова частота роботи мікроконтролера 20 МГц, а необхідна частота імпульсів 10 кГц то їх відношення буде 2000 або 11111010000, це 11 двійкових розрядів при максимальній можливій розрядності 10. Тому в цьому випадку розрядність ШІМ дорівнює 10. .[<http://picdevices.ru/eksperimentyi/pwm-dac-lpf-ili-kak-eto-po-nashemu-shim-tsap-fnch.html?doing_wp_cron=1464891333.4888598918914794921875>]

Забезпечення правильної роботи перетворювача не потребує конкретної частоти, тому було обрано максимальний період 255 та значення переддільника 1:1 в результаті буде отримано період:

Ppwm = (PR2 + 1) \* 4 \* Tosc \* Nprescale = 255 + 1 \* 4 1/(5 10е6) \* 1 = 20мкс = 0,2мс.

Це забезпечить досить високу частоту та максимальну розрядність ШІМ — 10 біт. При цьому значення шпаруватості може бути в діапазоні від 0 до 1023, отже можна змінювати вихідну напругу з кроком менше 0,1% від напруги живлення, що в даному випадку становить 0,05 В.

Отримання результатів вимірювання

Як відомо мікроконтролер є цифровим пристроєм та зазвичай працює з цифровими сигналами. Проте часто виникає необхідність отримувати дані з аналогових сигналів, як і в даному випадку. Не зважаючи на те, що існує велика кількість схем та інтегральних рішень аналогово-цифрових перетворювачів (АЦП) фірма Microchip намагається забезпечити якомога більше своїх мікроконтролерів вбудованим модулем АЦП, навіть найпростіші PIC10, що мають всього шість виводів.

Обраний мікроконтролер має 13 виводів (AN0–AN12) на які можна подавати аналогові сигнали. Однак від має лише один фізичний модуль АЦП, який підключається до цих виводів за допомогою мультиплексора.

Найважливішою характеристикою АЦП є його розрядність, вона показує з яким мінімальним кроком можна вимірювати величину аналогової напруги, що можна визначити за наступною формулою:

dU = Uadc / 2^Nrozr

де Uadc — опорна напруга АЦП

Nr — кількість розрядів АЦП.[<http://pro-diod.ru/programms/pic-micro/acp-v-pic18f45xx-analogo-cifrovoe-preobrazovanie.html>]

Обраний мікроконтролер має 10 розрядів АЦП, отже мінімальний крок аналогового сигналу при використанні напруги живлення як опорної становитиме приблизно 5 мВ.

Роботу модуля АЦП контролюють три восьми розрядні регістри ADCON0, ADCON1 та ADCON2. Перший дозволяє вимикати модуль та обирати який вивід буде підключено до нього. Другий встановлює джерело напруги та режими роботи виводів (аналоговий чи цифровий). Третій контролює час перетворення та частоту тактів модуля.[<http://ww1.microchip.com/downloads/en/DeviceDoc/39632e.pdf>]

Більшість бітів можна встановити як показано в демонстраційних прикладах, однак деякі потрібно розглянути детальніше.

Біт ADFM регістру ADCON2 визначає вирівнювання результату. Річ у тім, що 10 розрядів результату роботи АЦП знаходяться у 16-ти розрядному регістрі ADRES (ADRESH та ADRESL) і їх потрібно там певним чином розмістити (рис).

Рисунок

Якщо планується використовувати лише 8 розрядів результату, наприклад коли не потрібна велика точність вимірів або опорна напруга не досить стабільна рекомендується встановити лівостороннє вирівнювання (ADFM = 0), а результат брати з регістру ADRESH. Якщо ж планується використовувати всі 10 біт результату то краще вирівняти в право та скористатися ADRESH і ADRESL як одним 16-розрядним регістром.

Інша особливість використання модуля АЦП полягає в тому, що для своєї роботи він використовує переривання. Це потрібно, оскільки процес отримання цифрового значення сигналу займає певний час. В цей час мікроконтролер може виконувати інші дії, а коли перетворення буде завершено буде викликано переривання. Тож при ініціалізації модуля необхідно також настроїти переривання: біт ADIE регістру PIE1 дозволяє переривання від АЦП, біт ADIP регістру IPR1 встановлює пріоритет переривань від АЦП, біт ADIF регістру PIR1 скидає прапорець переривань.

Для запуску власне перетворення потрібно встановити біт GO\_DONE регістру ADCON0 та скинути ADIF, щоб переривання не було викликано одразу.

Щоб переконатись, що переривання викликане саме завершенням аналогово-цифрового перетворенням у функції обробнику потрібно перевірити, що встановлено одночасно ADIF та ADIE регістру PIE1, та скинути перший, що переривання не було викликано повторно.

Загальний алгоритм роботи мікроконтролера

Вище було описано налаштування та принципи роботи кожної необхідної частини окремо. А тут буде показано як вони взаємодіють в процесі вимірювання.

Під час запуску апаратної частини відбувається ініціалізація вказаних модулів з описаною конфігурацією.

При отриманні сигналу до початку вимірювань відбувається скидання всіх лічильників та встановлення значення напруги колектор-емітер і струму бази у початкове значення. Відбувається розрахунок початкового значення шпаруватості імпульсів для отримання необхідної напруги. Це значення контролюється за допомогою зворотнього зв’язку. Через резистивний дільник воно подається на вивід AN1. Коли значення напруги між двома запусками АЦП з даного виводу буде менше ніж задана похибка — вважається, що напруга встанвлена та починається вимірювання струму колектора. Після отримання цього значення напруга збільшується на заданий крок та все повторюється.

Кожна характеристика це досить великий об’єм даних. Максимальна розрядність ШІМ генератора 10, тобто можна отримати 1024 значення напруги. Розрядність керованого джерела струму 8, тобто 256 значень. Отже поверхня може містити

Нт = 2 Ншім \* 2 Нкдс = 1024 \* 246 = 262144 точок

Кожна точка точка представлена числом з плаваючою комою з одинарною точністью, тобто 4 байти на точку. Отже в результаті:

Спов = Сфлоат \* Нт = 4 \* 262144 = 1048576 байт = -= 1Мб

Обрана модель мікроконтролера має всього 2048 байт пам’яті даних, є неможливим збереження всіх результатів. Всі виміряні дані потрібно одразу ж предавати на ПК. Це можливо зважаючи на те, що всі використовувані модулі (таймери, АЦП та ШІМ) працюють у фоновому режимі і мікроконтролер може виконувати передачу на ПК між перериваннями.

Слід зазначити, що прийом та передача інформації на ПК має обмеження в часі, якщо дані небуде передано за відведений час то робота буже завершено з помилкою. Тому перед початком передачі потрібно бідкючити глобальні переривання (INTCON.GIE = 0), а після завершення дозволити.

Коли лічильник напруги досягне максимального значення він буде скинутий, а лічильник струму збільшений на заданий крок струму та процес повторено.

Для індикації стану приладу використано світлодіод, що буде блимати ти на протязі всього циклу вимірювань. Для цього використано апаратний таймер 0 мікроконтролера.