Микроконтроллеры и карты памяти Compact Flash

Олег Вальпа (Челябинская обл.)

В статье приводится краткий обзор карт памяти Compact Flash и способ их подключения к микроконтроллеру. Даётся описание библиотечных функций среды разработки mikroC для программной поддержки этих карт памяти. Приводится пример программы на языке высокого уровня С для работы микроконтроллера с ними.

Введение

С момента своего появления карты памяти привлекают внимание разработчиков микропроцессорной аппаратуры благодаря своей компактности, большому объёму памяти, малому потреблению энергии, надёжности и прочим преимуществам.

Применение данных носителей информации позволяет придать микроконтроллерным устройствам новые качества. Например, в этой памяти можно хранить набор рабочих программ, образ операционной системы, мультимедийные данные большого объёма и т.п. Хранимые в карте памяти программы и данные можно выбирать и загружать в микроконтроллер с помощью специально созданного программного меню. Кроме того, карта памяти позволит сохранять все необходимые в процессе работы устройства данные, обеспечивая их долговременное хранение в отключенном состоянии. Перенос данных можно осуществлять простым перемещением карты памяти и подключением её к другому устройству.

Рассмотрим один из представителей данных устройств памяти – карты памяти типа Compact Flash.

Карты памяти Compact Flash

Карты памяти Compact Flash (CF) представляют собой компактные накопители информации, базирующиеся на основе флэш-памяти. Большая ёмкость, высокая скорость записи данных и надёжность делают их очень привлекательными для использования в портативных устройствах. Внешний вид карты памяти CF приведён на рис. 1.

Рис. 1. Внешний вид карты памяти CF

Эти карты памяти появились на рынке компактных накопителей информации одними из первых. Формат данных карт памяти CF был разработан компанией San Disk Corporation в 1994 г. Спецификацию для данного формата поддерживает ассоциация Compact Flash (CFA). По мере развития технологий данный формат развивался, увеличивая объём и скорость обмена карт памяти. В настоящее время максимальный объём накопителей CF превысил сотню гигабайт, а скорость обмена достигла десятки Мб/с.

К большому достоинству карт CF относится наличие встроенного контроллера. Это обеспечивает самую широкую совместимость и стандартизацию. Карта CF поддерживает интерфейсный стандарт IDE ATA, и при считывании через специальное устройство компьютер видит его как обычный жёсткий диск без необходимости установки драйверов самой карты.

Потеря данных в CF предотвращается благодаря наличию специальной схемы обнаружения дефектов, а также технологии проверки и коррекции ошибок. Время хранения информации на карте может достигать 100 лет.

CF имеет прочную конструкцию и выдерживает большие ударные перегрузки. Размеры карт составляют 42 × × 36 мм, а толщина не превышает 5 мм. Благодаря своему размеру CF хорошо рассеивает тепло, что снижает вероятность перегрева.

Карты CF имеют 50-контактный разъём для сигнальных цепей интерфейса и цепей питания.

Скорость карты обычно обозначают как число и знак умножения «×». Число означает, во сколько раз скорость чтения карты больше, чем 150 Кб/с. Это минимальная, стандартная скорость чтения компакт-диска при минимальной стандартной скорости вращения шпинделя механизма привода. Данную скорость принято обозначать 1×. Например, обозначение скорости карты 40× соответствует 6 Мб/с, а 600× – 90 Мб/с.

Подключение карты к микроконтроллеру

Схема подключения карты CF к микроконтроллеру PIC18F452 семейства PIC компании MicroChip [1] приведена на рис. 2. Из этой схемы видно, что для подключения карты используется два восьмиразрядных порта микроконтроллера – PORTB и PORTD. Микроконтроллер тактируется с помощью кварцевого резонатора на частоте 8 МГц.

Программирование

Рассмотрим пример программы, которая осуществляет запись и чтение данных карты памяти. Для разработки таких программ вновь воспользуемся библиотекой функций microC [2]. Среда разработки microC предоставляет готовую библиотеку для работы микроконтроллеров с картами памяти CF.

В картах CF данные делятся на сектора, один сектор обычно состоит из 512 байт. Некоторые устаревшие модели имеют сектора по 256 байт. Операции чтения и записи с помощью библиотечных функций осуществляются не напрямую, а последовательно, через буфер на 512 байт. Программы библиотеки могут быть использованы для работы с картами CF, имеющими файловые системы FAT16 и FAT32. Однако программы управления файлами могут работать только с файловой системой FAT16.

Программы доступа к файлам позволяют осуществлять запись и чте-

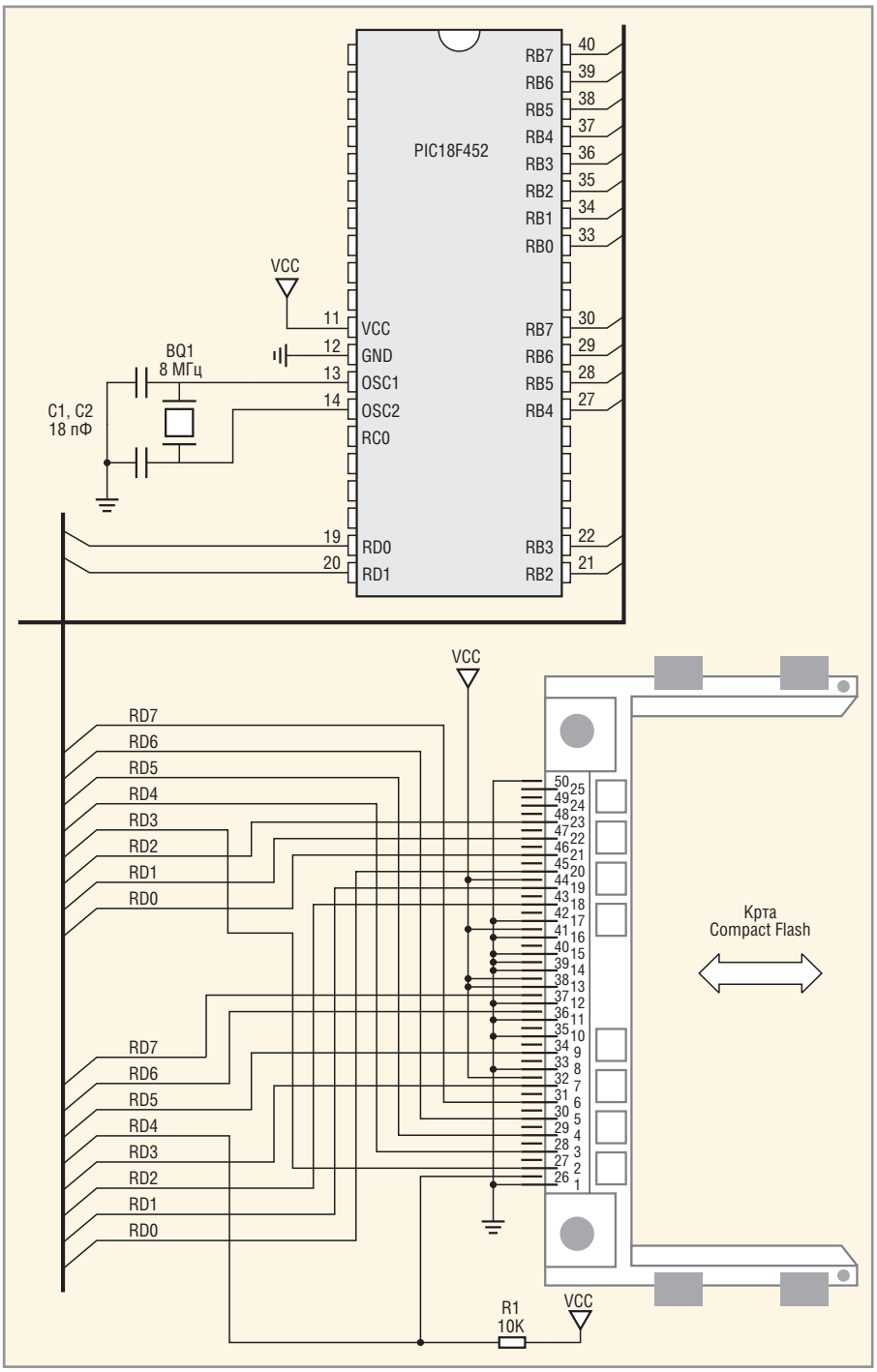


Рис. 2. Схема подключения карты памяти CF к микроконтроллеру

ние файлов. Имя файла должно быть длиной ровно 8 символов верхнего регистра. Пользователь сам отвечает за уникальность имён файлов, т.к. программы работы с CF этого не контролируют.

Перед операцией записи необходимо убедиться, что запись производится не в загрузочный сектор или сектор таблицы FAT, т.к. это сделает карту нечитаемой на компьютере или в цифровой камере. Здесь существенную помощь могут оказать средства низкоуровневого доступа к носителям, вроде программы Winhex.

Библиотека среды разработки mikroC содержит следующий набор функций для работы с картами памяти CF:

- Cf_Init,
- Cf_Detect,
- Cf_Total_Size,
- Cf_Enable,
- Cf_Disable,
- Cf_Read_Init,
- Cf_Read_Byte,
- Cf_Write_Init,
- Cf_Write_Byte.

Описание данных функций, требований к ним и пример их использования приводится в табл. 1–9.

Кроме того, библиотека mikroC содержит набор функций для работы с файлами. Данный набор включает в себя следующие функции:

- Cf_Fat_Init,
- Cf_Fat_Assign,
- Cf_Fat_Reset,
- Cf_Fat_Read,
- Cf_Fat_Rewrite,
- Cf_Fat_Append,
- Cf_Fat_Delete,
- Cf_Fat_Write,
- Cf_Fat_Set_File_Date,
- Cf_Fat_Get_File_Date,
- Cf_Fat_Get_File_Size.

Описание этих функций приведено в табл. 10–20.

Пример использования библиотечных функций

Следующий пример демонстрирует использование библиотечных функций при работе с картами памяти CF. В данном примере осуществляется инициализация портов микроконтроллера и организуется запись 512 байт в сектор 590, затем программа читает данные и выводит их в PORTD для визуального контроля:

```
unsigned i;
void main() {
TRISC = 0; // PORTC - выход
Cf_Init(&PORTB, &PORTD); // Инициализация портов
while (!Cf_Detect()); // Ожидание подключения CF карты
Delay_ms(500);
Cf_Write_Init(590, 1); // Инициализация записи сектора 590
// Запись 512 байтов в сектор (590)
```

Таблица 1. Описание функции Cf_Init

Прототип	void Cf_Init(unsigned short *ctrlport, unsigned short *dataport);
Возвращаемое значение	Нет
Описание	Инициализирует порты для работы с CF-картой. Определяет два порта: ctrlport и dataport
Требования	Нет
Пример	Cf_Init(&PORTB, &PORTD);

Таблица 2. Описание функции Cf_Detect

Прототип	unsigned short Cf_Detect(void);
Возвращаемое значение	Возвращает 1, если CF карта присутствует, в противном случае возвращает 0
Описание	Определяет наличие CF-карты на порте ctrlport
Требования	Порт управления (ctrlport) должен быть проинициализирован с помощью функции Cf_Init
Пример	Ожидание, пока не будет вставлена CF карта: do nop; while (!Cf_Detect());

Таблица 3. Описание функции Cf_Total_Size

Прототип	unsigned long Cf_Total_Size(void);
Возвращаемое значение	Размер карты в килобайтах
Описание	Возвращает размер CF-карты в килобайтах
Требования	Порт должен быть проинициализирован. См. Cf_Init
Пример	size = Cf_Total_Size();

Таблица 4. Описание функции Cf_Enable

Прототип	void Cf_Enable(void);
Возвращаемое значение	Нет
Описание	Разрешение устройства. Программу придётся вызывать, только если предварительно карта была запрещена с помощью программы Cf_Disable. Эти две функции вместе позволяют занимать или освобождать линии данных при работе одновременно с несколькими устройствами. См. пример в конце главы
Требования	Порт должен быть проинициализирован. См. Cf_Init
Пример	Cf_Enable();

Таблица 5. Описание функции Cf_Disable

Прототип	void Cf_Disable(void);
Возвращаемое значение	Нет
Описание	Программа запрещает устройство, освобождая линии данных для других устройств. Чтобы вновь разрешить работу с устройством, следует вызвать Cf_Enable. Эти две функции вместе позволяют занимать или освобождать линии данных при работе одновременно с несколькими устройствами. См. пример в конце главы
Требования	Порт должен быть проинициализирован. См. Cf_Init
Пример	Cf_Disable();

Таблица 6. Описание функции Cf_Read_Init

Прототип	void Cf_Read_Init(unsigned long address, unsigned short sectcnt);
Возвращаемое значение	Нет
Описание	Инициализирует CF-карту для чтения. Параметр address определяет адрес сектора, откуда будут читаться данные, а sectcnt – это полное количество секторов, подготовленных к операции чтения
Требования	Порт должен быть проинициализирован. См. Cf_Init
Пример	Cf_Read_Init(590, 1);

Таблица 7. Описание функции Cf_Read_Byte

Прототип	unsigned short Cf_Read_Byte(void);
Возвращаемое значение	Возвращает байт из CF
Описание	Читает один байт из CF
Требования	Порт должен быть проинициализирован. См. Cf_Init Карта памяти CF должна быть проинициализирована для чтения. См. Cf_Read_Init
Пример	Чтение байта и вывод его в PORTC: PORTC = Cf_Read_Byte();

Таблица 8. Описание функции Cf_Write_Init

Прототип	void Cf_Write_Init(unsigned long address, unsigned short sectcnt);
Возвращаемое значение	Нет
Описание	Инициализирует CF-карту для записи. Параметр address определяет адрес сектора, куда будут записываться данные, а sectcnt – это полное количество секторов, подготовленных для записи
Требования	Порт должен быть проинициализирован. См. Cf_Init
Пример	Cf_Write_Init(590, 1);

Таблица 9. Описание функции Cf_Write_Byte

Прототип	void Cf_Write_Byte(unsigned short data);
Возвращаемое значение	Нет
Описание	Записывает 1 байт (data) в буфер CF. Все 512 байтов передаются в буфер. Пока буфер на 512 байтов не заполнен – данные не записываются в CF
Требования	Порт должен быть проинициализирован. См. Cf_Init. CF должна быть инициализирована для записи. См. Cf_Write_Init
Пример	Cf_Write_Byte(100);

Таблица 10. Описание функции Cf_Fat_Init

Прототип	unsigned short Cf_Fat_Init(char * ctrlport, char * dataport);
Возвращаемое значение	Возвращает 0, если инициализация была успешной, 1 – если загрузочный сектор не был найден, и 255 – если карта не была обнаружена
Описание	Инициализирует порты для работы с FAT-системой CF-карты. Определяет два различных порта: ctrlport и dataport
Требования	Нет
Пример	CF_Fat_Init(PORTD,PORTC);

Таблица 11. Описание функции Cf_Fat_Assign

Прототип	unsigned short Cf_Fat_Assign(char *filename, char create_file);
Возвращаемое значение	1, если файл присутствует (или файл отсутствует, но создаётся новый файл), или 0, если файл отсутствует и новый файл не создаётся
Описание	Назначает файл для операций с FAT-системой. Если такой файл отсутствует, функция может создать новый файл с таким именем. Параметр filename – это имя файла (должно быть в формате 8.3 и в верхнем регистре). Параметр create_file – это признак создания новых файлов. Если create_file отличается от 0, то при отсутствии файла с таким именем создаётся новый файл
Требования	Порты должны быть инициализированы для операций с FAT-системой карты. См. Cf_Fat_Init
Пример	Cf_Fat_Assign('MIKROELE.TXT',1);

Таблица 12. Описание функции Cf_Fat_Reset

Прототип	void Cf_fat_Reset(unsigned long *size);
Возвращаемое значение	Размер файла в байтах. Размер сохраняется в адресе, заданном аргументом
Описание	Открывает назначенный файл для чтения
Требования	Порты должны быть инициализированы для операций с FAT-системой карты. См. Cf_Fat_Init Файл должен быть назначен. См. Cf_Fat_Assign
Пример	Cf_Fat_Reset(size);

Таблица 13. Описание функции Cf_Fat_Read

Прототип	void Cf_Fat_Read(unsigned short *bdata);
Возвращаемое значение	Нет
Описание	Читает данные из файла. bdata – данные, прочитанные из файла
Требования	Порты должны быть инициализированы для операций с FAT-системой карты. См. Cf_Fat_Init Файл должен быть назначен. См. Cf_Fat_Assign Файл должен быть открыт для чтения. См. Cf_Fat_Reset
Пример	Cf_Fat_Read(character);

Таблица 14. Описание функции Cf_Fat_Rewrite

Прототип	void Cf_fat_Rewrite();
Возвращаемое значение	Нет
Описание	Открывает новый файл для записи
Требования	Порты должны быть инициализированы для операций с FAT-системой карты. См. Cf_Fat_Init Файл должен быть назначен. См. Cf_Fat_Assign
Пример	Cf_Fat_Rewrite();

Таблица 15. Описание функции Cf_Fat_Append

Прототип	void Cf_fat_Append();
Возвращаемое значение	Нет
Описание	Открывает существующий файл для записи. Новые данные будут добавляться в конец файла
Требования	Порты должны быть инициализированы для операций с FAT-системой карты. См. Cf_Fat_Init Файл должен быть назначен. См. Cf_Fat_Assign
Пример	Cf_Fat_Append();

Таблица 16. Описание функции Cf_Fat_Delete

Прототип	void Cf_Fat_Delete();
Возвращаемое значение	Нет
Описание	Удаляет файл из CF
Требования	Порты должны быть инициализированы для операций с FAT-системой карты. См. Cf_Fat_Init Файл должен быть назначен. См. Cf_Fat_Assign
Пример	Cf_Fat_Delete();

Таблица 17. Описание функции Cf_Fat_Write

Прототип	void Cf_fat_Write(char *fdata, unsigned data_len);
Возвращаемое значение	Нет
Описание	Записывает данные в файл на CF. Аргумент fdata – данные, записываемые в файл на CF. Аргумент data_len – количество байтов, записываемых в файл на CF
Требования	Порты должны быть инициализированы для операций с FAT-системой карты. См. Cf_Fat_Init Файл должен быть назначен. См. Cf_Fat_Assign Файл должен быть открыт для записи. См. Cf_Fat_Rewrite или Cf_Fat_Append

Пример	// записать данные в назначенный файл
	Cf_Fat_Write(file_contents, 42);

Таблица 18. Описание функции Cf_Fat_Set_File_Date

Прототип	void Cf_fat_Set_File_Date(unsigned int year, unsigned short month, unsigned short day, unsigned short hours, unsigned short mins, unsigned short seconds);
Возвращаемое значение	Нет
Описание	Установка атрибутов времени файла. Можно задавать год, месяц, день, часы, минуты и секунды
Требования	Порты должны быть инициализированы для операций с FAT-системой карты. См. Cf_Fat_Init Файл должен быть назначен. См. Cf_Fat_Assign Файл должен быть открыт для записи. См. Cf_Fat_Rewrite or Cf_Fat_Append
Пример	Cf_Fat_Set_File_Date(2005,9,30,17,41,0);

Таблица 19. Описание функции Cf_Fat_Get_File_Date

Прототип	void Cf_fat_Get_File_Date(unsigned int *year, unsigned short *month, unsigned short *day, unsigned short *hours, unsigned short *mins);
Возвращаемое значение	Нет
Описание	Читает атрибуты времени файла. Можно прочитать year, month, day, hours, mins
Требования	Порты должны быть инициализированы для операций с FAT-системой карты. См. Cf_Fat_Init Файл должен быть назначен. См. Cf_Fat_Assign
Пример	Cf_Fat_Get_File_Date(year, month, day, hours, mins);

Таблица 20. Описание функции Cf_Fat_Get_File_Size

Прототип	unsigned long Cf_fat_Get_File_Size();
Возвращаемое значение	Размер файла в байтах
Описание	Функция возвращает значение размера файла в байтах
Требования	Порты должны быть инициализированы для операций с FAT-системой карты. См. Cf_Fat_Init Файл должен быть назначен. См. Cf_Fat_Assign
Пример	Cf_Fat_Get_File_Size;

```
for (i = 0; i < 512; i++)
Cf_Write_Byte(i + 11);
PORTC = 0xFF;
Delay_ms(1000);
Cf_Read_Init(590, 1); // Инициа-
лизация чтения сектора 590
// Чтение 512 байтов из сектора
(590)
for (i = 0; i < 512; i++) {
PORTC = Cf_Read_Byte(); // Чтение
байта и вывод его в PORTC
Delay_ms(1000);
}
}
```

Можно использовать приведённую здесь программу в качестве примера организации общения микроконтроллера с картой памяти CF и создавать на её основе программы специального назначения.

Заключение

Аналогично рассмотренному здесь варианту использования микроконтроллера семейства PIC, можно применять для работы с картами памяти CF и микроконтроллеры других семейств. В частности, существуют среды разработки для поддержки микроконтроллеров семейства 8051 и микроконтроллеров семейства AVR [3]. Данные среды разработки имеют идентичные библиотечные функции для работы с CF.

Литература

1. www.microchip.com.
2. www. mikroe.com.
3. *Вальпа О.* Современная среда разработки mikroC для программирования микроконтроллеров на языке высокого уровня Си. Современная электроника. 2010. № 6. С. 64.

Новости мира News of the World Новости мира

Материал для солнечных батарей будущего

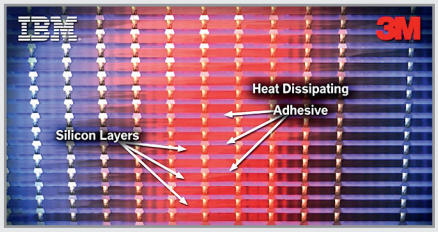
Группа исследователей из Гарвардского университета и других научных центров разработала методику для автоматизированного изучения молекулярных структур с целью создания органических фотогальванических приборов. В основе лежит идея создания высокоэффективных солнечных батарей на основе органики, которые бы выдавали энергию, сопоставимую по стоимости с традиционными источниками. Цель проекта Clean Energy Project – выявить из множества органических материалов такие, с помощью которых можно будет создать солнечные батареи с КПД 10…15% и сроком службы более 10 лет. Помимо традиционных методов анализа, участники проекта надеются привлечь добровольцев в организованную ими сеть распределённых вычислений, напоминающую известный проект SETI.

Согласно предварительному анализу, лишь 0,3% органических структур (3000 – 5000) из содержащихся в библиотеке проекта могут иметь необходимый энергетический уровень для создания элементов с эффективностью преобразования 10% и более. Использование традиционных подходов к анализу и синтезу таких структур с учётом накопленного опыта и эмпирической интуиции позволяет исследовать лишь несколько вариантов в год. В то же время, как говорит один из лидеров проекта Алан Аспуру-Гузик (Alan Aspuru-Guzik), каждые 12 ч «донорского» процессорного времени позволяют выделить одну кандидатуру для более пристального изучения структуры и свойств материала. Ближайшая цель проекта как раз и состоит в создании базы таких веществ для дальнейшего изучения. Методика поиска вероятных кандидатов из более чем 10-миллионного списка сочетает в себе как обычные стратегии моделирования, так и алгоритмы синтеза современных лекарственных препаратов в сочетании с идеями распознавания образов и компьютерной химии. Таким образом, каждый пользователь ПК, имеющий доступ в Сеть, может скачать программу с сайта и участвовать в проекте, выделяя часть вычислительных мощностей своего компьютера для поиска перспективного материала.

http://cleanenergy.harvard.edu/

IBM и 3M разрабатывают клей для полупроводников

Компьютерный гигант IBM и известная своими инновационными разработками компания 3M объявили о сотрудничестве в создании специального клея для полупроводников. Ключевой особенностью нового



вещества будет возможность создавать многослойные полупроводниковые структуры, что позволит существенно уменьшить размеры интегральных схем.

По словам сотрудников IBM, этот клей позволит создавать структуры, содержащие до 100 слоев полупроводников. Такое решение будет применяться для более тесной интеграции при создании SoC. Так, учёные уже видят всего одну микросхему, которая будет в себе объединять вычислительный процессор, память и сетевые функции.

Стоит сказать, что инженеры IBM делали попытки продвинуться в этом направлении и ранее, но без подобных материалов у них ничего не получалось. Построенные таким образом чипы будут использоваться в портативной электронике, в частности в планшетах, так как для них размер имеет первоочередное значение. Как сама разработка, так и первые решения на её базе будут готовы к 2013 г. К сожалению, пока непонятно, как инженеры собираются бороться с выделением тепла, но думается, что у них есть соображения на этот счёт.

http://www.cnet.com/