Flash-Controller mit Debugger on chip

Neue Anwendungen mit Flash-Mikrocontroller

Rodger L. Richey ***Anfangs waren Flash-Mikrocontroller ein praktisches Entwicklungswerkzeug für Designer, entfiel doch die lästige, typische 20-minütige Wartezeit für das UV-Löschen des Fenster-EPROMs. Die Flash-Speichertechnik hat sich seit diesen Tagen weiterentwickelt; heute gibt es nicht nur kostengünstige Bauteile mit kurzen Programmierzeiten, sondern auch erweiterte Funktionen, die selbstprogrammierbare Anwendungen ermöglichen und selbst den Mikrocontroller auch als Debugger nutzbar machen. Der Artikel erläutert die verschiedenen Aspekte der Entwicklungsarbeit mit Flash-Speichern und beschreibt eine Anwendung, welche diese neue Technik nutzt.***

Die wohl wichtigste Funktion des Flash-Controllers ist der Programmiermechanismus für den internen Programmspeicher. Die meisten, wenn nicht sogar alle Flash-Mikrocontroller bieten einen externen Programmiermodus über eine parallele oder serielle Schnittstelle. Mit einem parallelen Programmiermodus kann man die Programmierzeit erheblich verkürzen; eine serielle Schnittstelle eröffnet demgegenüber ein unaufwendiges Programmieren des Bauteils im eingebauten Zustand in der Anwendungsschaltung. Ein Beispiel dafür ist die ICSP-Schnittstelle (In-Circuit Serial Programming), die z. B. in den meisten PICmicro-Mikrocontrollern von *Microchip Technology* enthalten ist. Diese Schnittstelle bildet mit nur zwei I/O-Pins eine serielle Schnittstelle zwischen Programmer und Bauteil. Ein Pin arbeitet als serieller Clock vom Programmer zum Bauteil, das andere als bidirektionale Datenleitung. Die Programmierschnittstelle umfasst außerdem Power-, Masse- und Reset-Pins am Mikrocontroller.

Entwickeln mit Flash-Memories

Nach der Schnittstellendefinition muss der Entwickler an die Programmierspannungen zur Programmierung des Flash-Speichers denken. Manche Bauteile benötigen Spannungen, die größer sind als die Betriebsspannung der Anwendung. Andere Bauteile lassen sich bei 5 V programmieren, so dass keine zusätzlichen Überspannungsschaltungen nötig sind. Obwohl sie vielseitiger ist als die Überspannungs-Programmierlösung, engt die 5-V-Programmierlösung den Betrieb bei niedrigen Spannungen noch immer ein. Für beste Flexibilität sollte eine Programmierung über den gesamten Betriebsspannungsbereich des Bauteils möglich sein. Auch hier bietet das ICSP-Protokoll der 8-bit-Flash-Mikrocontrollerfamilie PIC16F87X ein Höchstmaß an Flexibilität. In der Überspannungs-ICSP-Betriebsart wird eine 13-V-Versorgungsspannung zum Programmieren und eine V_{dd}-Spannung von 5 V benötigt. Das Bauteil unterstützt auch den 5-V-Programmiermodus. PIC16F87X-Bauteile gehen aber noch einen Schritt weiter: Sie lassen sich bei jeder beliebigen Betriebsspannung zwischen 2,0 V und 5,5 V wortbezogen reprogrammieren.

Selbstprogrammierung

Höchste Flexibilität bieten Mikrocontroller, die sich selbst programmieren können: Der Mikrocontroller nimmt Daten aus einer externen Quelle über eine Peripherieschaltung wie z.B. ein USART entgegen. Er schreibt diese Daten dann ohne irgendwelchen externen Spannungen in seinen eigenen Programmspeicher. Am einfachsten schließt man dazu den Mikrocontroller (mit passender Pegelumsetzung) über den RS-232 Port direkt an einen PC an. Sobald das Programm assembliert oder compiliert ist, kann man die HEX-Ausgabedatei in den Mikrocontroller über den auf der Baugruppe integrierten USART einspie-

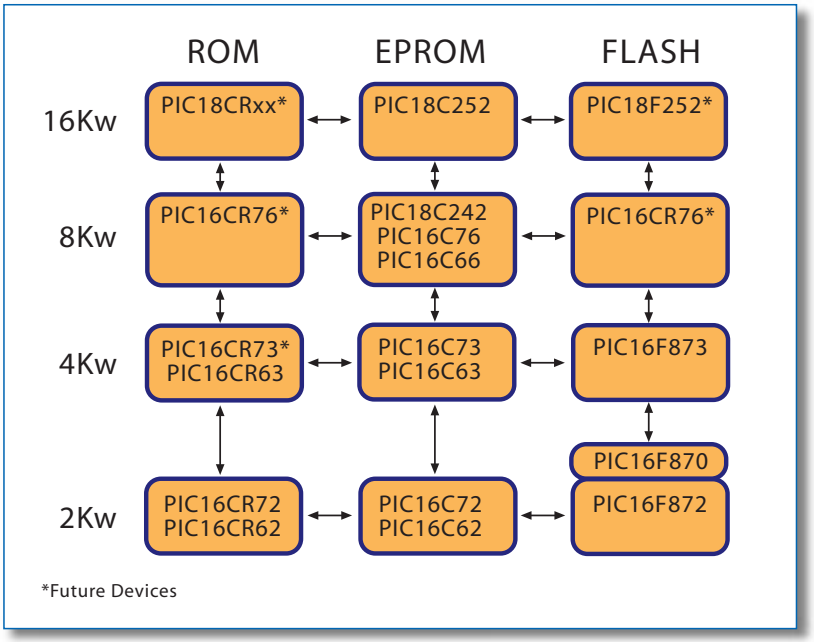

Bild 1: Speicher-Wachstumspfade für 28-polige Mikrocontroller

len. Das auf dem Mikrocontroller laufende Bootloader-Programm kann dann den ankommenden Datenstrom in Adressen und Daten auftrennen und anschließend in den Speicher schreiben. Der Mikrocontroller sollte diese Schreibvorgänge annehmen, sobald diese die Betriebs-, Frequenz- und Spannungsbereiche des Bauteils angeben.

Anwendungen

Eine batteriebetriebene Embedded-Anwendung mit Telefonzugang kann dieses Konzept optimal nutzen. Über ein TCP/IP-Stack-Chip (z.B. *Seiko S-7600A*) und ein integriertes Modem kann der Mikrocontroller einen Internet Service Provider anwählen und eine Client/Server-Verbindung zu einer Host-Maschine herstellen. Der Mikrocontroller kann Daten zum Server hochladen, und der Server kann auch neue Codeabschnitte in den Mikrocontroller herunterladen.
Diese neuen Programmierfunktionen eines Flash-basierten Mikrocontrollers sind in vielen Anwendungen nützlich.
Eine weitere Möglichkeit ist ein Low-Cost Entwicklungswerkzeug, der MP LAB-ICD In-Circuit Debugger von Microchip, das sich zum Debuggen einer Anwendung während der Prototypen-Phase eignet. Dieses Werkzeug nutzt die serielle Programmierschnittstelle zur Kommunikation mit der Entwicklungsumgebung. Der Flash-Programmspeicher ermöglicht die Entwicklung von neuem Code, der sofort in das Bauteil heruntergeladen werden kann. Dieses Werkzeug erfüllt die Grundanforderungen eines Debugging-Werkzeugs: Breakpoints, Watch-Windows usw. Der einzige Nachteil des Werkzeugs ist, dass die Debug-Funktionen Platz auf dem Die belegen, der sich nicht für die Schaltung nutzen lässt. Mit diesem Werkzeug lassen sich auf PIC16F87X-Bauteilen kostengünstig einfache Debugfunktionen realisieren. Sie nutzen die komfortable integrierte MPLAB-Entwicklungsumgebung und einen einzigen RS-232 Seriell-Port auf einem PC. Dieses Werkzeug bietet viele wichtige Merkmale, die selbst moderne In-Circuit Emulatoren heute nicht bieten können.
Ein weiterer Vorteil des MPLAB-ICD Werkzeugs ist, dass man jetzt SMD-Gehäuse mit Fine-Lead-Pitch und Mikrocontroller in Die-Form über den Flash-Mikrocontroller mit Debugger emulieren kann. Schon seit langem konnte man In-Circuit Emulatoren bei Anwendungen mit Fine-Lead-Pitch-Gehäusen einsetzen, die dafür erforderlichen Adapter waren aber sehr teuer. Nur sehr selten emulierte man Die-Anwendungen, weil das Anbringen der Clip-Verbindungsleitungen an Bonddrähten schwierig ist. Neue technologische Entwicklungen bei Wire-Bonding-Systemen und Kostensenkungen haben aber neue Märkte geschaffen, so dass Die-Anwendungen immer häufiger eingesetzt werden. Der Einsatz von Dies ermöglicht weitere Kostensenkungen, weil die Kosten für Gehäuse entfallen.

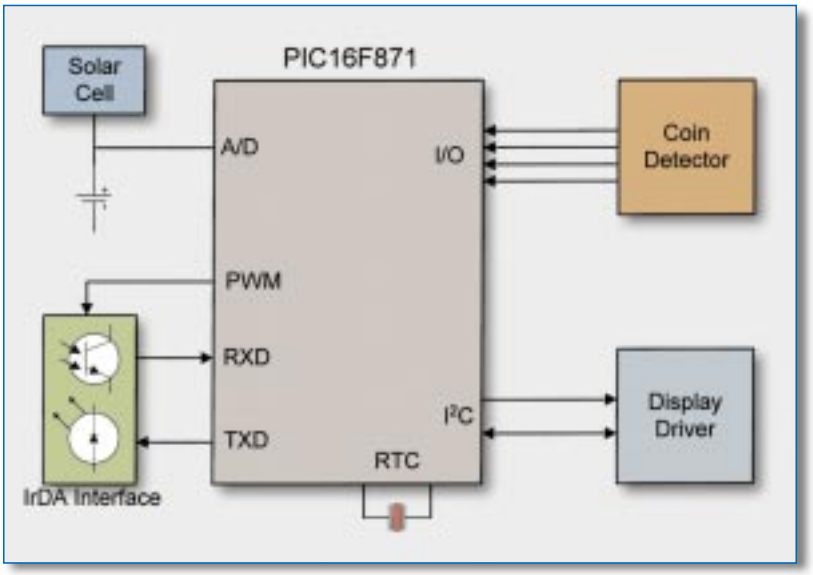

***Bild 2*: Parkuhr-Blockdiagramm**

Der Controller als Debugger

Der nächste wichtige Punkt des Entwicklungswerkzeugs: Jeder Mikrocontroller lässt sich auch als Debugger nutzen. Der Debugger-Einsatz bringt Vorteile für die Anwendung, da das echte Bauteil in der Schaltung verwendet wird. Der „echte" Oszillator und die „echten" I/O-Pins interagieren mit der Anwendung. In-Circuit Emulatoren besitzen typischerweise Schutzbausteine an den I/O-Pins und können zahlreiche Verbindungen zwischen der emulierenden Schaltung und der Verbindung zur Schaltung besitzen. Die Anwendung wird über geringere Treiberstärken und eine ungewöhnliche Belastung der I/O-Pins beeinträchtigt, was besonders bei Analogkanälen (AD-Wandler und Komparatoren) zu Störungen führt. Ein Debugger bietet das beste aus beiden Welten: den Betrieb mit realen Bauteilen samt Debugfunktionen.

Ein Debugger ist auch dann nützlich, wenn eine Mikrocontroller-Produktfamilie ähnliche Pinouts und Funktionsmerkmale besitzt. Ein Debugger, der auf einem Mikrocontroller läuft, lässt sich zur Emulation von Bauteilen mit anderen Speicher-Technologien wie ROM oder EPROM nutzen. **Bild 1** illustriert die *Migratable Memory* Technologie, die Sockel- und Software-Kompatibilität zwischen allen gleichwertigen ROM-, OTP- und Flash-Mikrocontrollern der PICmicro-Produktfamilie von Microchip bietet. Somit können Anwender die Mikrocontroller-Speichertechnologie und Speichergröße an die Produktlebensdauer ihrer Anwendung anpassen, und bei Bedarf auf eine kostengünstigere Lösung umsteigen.

Unkonventionell: Flash in der Parkuhr

Die Flash-Technik kann den Funktionsumfang vieler nicht-traditioneller Anwendungen wie z.B. den einer Parkuhr verbessern. Früher waren Parkuhren rein mechanische Geräte. Man wirft ein Paar Münzen ein, dreht einen Knopf, und bezahlt damit einen Parkplatz für eine gewisse Zeit. Die Parkuhr kennt weder Uhrzeit noch Wochentag. Sie zählt lediglich die bezahlte Zeit herunter. In manchen Städten wären je nach Tag und Uhrzeit unterschiedliche Parkgebühren vorteilhaft. Außerdem könnte die örtliche Polizei je nach der vergangenen Zeit seit dem Ablauf der Parkzeit entsprechende Verwarngebühren verhängen. Die hier vorgestellte Parkuhr-Anwendung beruht auf einem PIC16 F871 8-bit Mikrocontroller. Das Bauteil enthält zahlreiche Peripheriefunktionen und einen Flash-Programmspeicher, der die Entwicklung vereinfacht und eine kostengünstige Lösung ermöglicht. **Bild 2** zeigt das Blockdiagramm der Parkuhr-Anwendung mit einem PIC16F871.
Das erste Problem für die Anwendung ergibt sich daraus, dass die meisten Parkuhren mechanisch und nicht elektrisch

arbeiten. So kommt zwingend als Stromquelle ein Akku zum Einsatz. Dieser Akku muss entsprechend überwacht und geladen werden. Am besten verwendet man dazu eine Solarzelle, die den Akku untertags kontinuierlich lädt. Die Ladesteuerung ist in den Mikrocontroller eingebaut und verwendet einen auf der Baugruppe integrierten 10-bit-AD-Wandler sowie ein 10-bit-PWM-Modul. Der AD-Wandler überwacht die Batteriespannung, der PWM steuert Ladespannung/-Strom für den Akku. Der Flash-Programmspeicher kommt als Erstes beim Akku-Ladealgorithmus zum Einsatz. Da sich Akku-Technologien noch stetig weiterentwickeln, könnte die Parkuhr neue Akkus nutzen, die im praktischen Einsatz einen anderen Algorithmus benötigen. Der aktuelle Ladealgorithmus im Flash-Programmspeicher kann per Reprogrammierung vor Ort durch einen neuen ersetzt werden.

Datenschnittstelle

Die zweite Anwendungsanforderung ist das Verfolgen der Zeit - nicht der bezahlten Zeit, sondern der Uhrzeit und der verstrichenen Zeit nach Ablauf der Parkzeit. Der PIC16F871 besitzt einen Timer, der asynchron zum Master-Clock arbeiten kann und für den Betrieb mit 32-kHz-Uhrenquarzen ausgelegt ist. Dieser Timer wird als Echtzeituhr verwendet und verwaltet den Countdown der Parkzeit sowie die Zeit nach Parkzeit-Ablauf.
Eine der wichtigsten Funktionen der Parkuhr ist die Datenschnittstelle, über die man Statusinformationen auslesen und neue Daten in die Parkuhr laden kann. Der Polizeibeamte könnte sich dafür interessieren, wie lange die Parkuhr bereits abgelaufen ist, um eine entsprechende Verwarnung auszustellen. Ein Techniker könnte neue Parkgebühren in die Parkuhr einprogrammieren. Die beste, am effektivsten gegen Manipulationen geschützte Schnittstelle ist ein optisches Interface. Das USART-Modul im PIC16F871 lässt sich über ein weiteres PWM-Modul an ein Low-Speed IrDA-Chipset anschließen. Das PWM liefert den 16x Clock für das IRDA-Chipset; das USART lässt sich auf eine beliebige Baudrate programmieren. Viele Laptops oder Taschencomputer wie der PalmPilot besitzen IRDA-Schnittstellen, die sich für die Kommunikation mit der Parkuhr eignen. Zur Statusüberwachung der Parkuhr und zum Herunterladen neuer Daten könnte man ein einfaches Programm schreiben, das auf diesen Geräten läuft.
Die weiteren Parkuhrfunktionen, die der PIC16F871 übernehmen muss, sind der Münzdetektor und der Display-Treiber. Der PIC16F871 besitzt 33 I/O-Pins, an die sich leicht eine Münzwaage anschließen lässt. Das beste Display für eine Parkuhr ist wohl eine LCD-Anzeige. Sie bietet besten Kontrast unter allen Beleuchtungsbedingungen und benötigt im Vergleich mit LED-Displays sehr wenig Strom. Die vielen andere Interface-Module im PIC16F871, wie z.B. SPI oder I^2C lassen sich für den Anschluss eines LCD-Display-Treibers verwenden.
Schließlich kommt noch eine Zusatzfunktion der Parkuhr, die Schutz bei der Geldleerung bietet. Diebstahl durch Mitarbeiter bei der Münzentleerung kann ein ernstes Problem sein. Der integrierte EEPROM-Datenspeicher läßt sich zur Speicherung des Geldbetrags oder der Münzzahl nutzen, die seit der letzten Leerung eingeworfen wurden. Mechanische Parkuhren können die Zahl der eingeworfenen Münzen oder den Betrag des eingeworfenen Geldes nicht erfassen. Dieser Wert lässt sich im EEPROM-Datenspeicher aufzeichnen und bei jeder Wartung der Parkuhr in einen Taschencomputer auslesen. Diese Rechner liefern die Seriennummer der Parkuhr und den eingenommenen Geldbetrag, die man dann am Ende des Tages überprüfen kann. Unstimmigkeiten lassen sich durch einen Vergleich mit den Computerdaten lösen, denn Computer täuschen sich bekanntlich nur sehr selten.

Flash-Mikrocontroller entwickeln sich zur Technologie der Wahl für die meisten Entwickler. Fortschritte bei der Fertigung Flash-basierter Bauteile ermöglichen auch in Zukunft sinkende Flash-Kosten, was neue, innovative Anwendungen eröffnet. Entscheidend für die Auswahl eines geeigneten Mikrocontrollers sind die Art der Funktionen, die Flash-Speicher bieten: die verfügbaren Programmiermodi und Debugging-funktionen. Eine weiterer wichtiger, nicht mit Flash-Speichern verbundener Faktor ist der auf- und abwärtsgerichtete Wachstumspfad, den eine Mikrocontroller-Familie bieten kann. Oft sind Flash-Speicher optimal für die Entwicklung, für den Feldeinsatz aber wenig sinnvoll. EPROM- und ROM-Versionen eröffnen hier eine weitere Dimension an Kosteneffizienz und Flexibilität.

Rodger L. Richey *ist Applications Project Manager bei Microchip Technology Inc.*