AN037

Application Note zu Multi-LAB/2

Initialisierung einer Multi-LAB/2-Karte ohne die mitgelieferten PC-Bibliotheken

Autor: MIH AN037.DOC (10 Seiten)

INHALT:

1. Definitionen 2

2. Aufgabenstellung 2

3. Der Plug-and-Play-Mechanismus der Multi-LAB/2-Karte 3

4. Die Steuerung des lokalen Prozessors 3

5. Vorgehensweise beim Laden eines Betriebssystems 4

5.1 Prüfen des Zugriffsmodus 4
5.2 Rücksetzen des Befehlszeigers 4
5.3 Setzen des Datenzeigers und Schreiben von Daten 5
5.4 Starten eines Programms 5
5.5 Laden und Starten des Betriebssystems OsX 5

6. Listings 6

6.1 Allgemeine Bemerkungen zu den Listings 6
6.2 Quellcode 7

Hinweis: Die Angaben der vorliegenden Application Note sind ohne Gewähr. SORCUS behält sich technische Änderungen vor.

1. Definitionen

Im folgenden werden einige für das Verständnis wesentliche Begriffe und Kürzel erklärt:

Basisadresse:	PC-I/O-Adresse, unter der die Multi-LAB/2-Karte installiert ist. Sie ist die erste der acht aufeinanderfolgenden PC-I/O-Adressen, die eine Multi-LAB/2-Karte verwendet.
Statusregister:	Dieses Register ist unter der Basisadresse + 0 auslesbar. Die Bedeutung der einzelnen Bits (DLP, RBF, DLM, TBF) ist dem Benutzer-Handbuch der Multi-LAB/2-Karte zu entnehmen.
lokaler Prozessor:	Prozessor der Multi-LAB/2-Karte
lokaler Speicher:	Speicher der Multi-LAB/2-Karte
lokale Adresse:	eine Adresse im lokalen Speicher
IP:	Befehlszeiger des lokalen Prozessors
Datenzeiger:	Zeiger auf den Speicher der Multi-LAB/2-Karte
Ladeadresse:	Adresse im Speicher der Multi-LAB/2-Karte, ab der das Betriebssystem geladen wird.
Startadresse:	Adresse im Speicher der Multi-LAB/2-Karte, ab der das Betriebssystem gestartet wird.
Kartennummer:	Werden mehrere Multi-LAB/2-Karten in einem Rechner verwendet, so müssen diese unterschiedliche Nummern besitzen. Die Kartennummer wird mithilfe eines Drehschalters auf der Multi-LAB/2-Karte eingestellt.

2. Aufgabenstellung

Aufgrund des jumperfreien Designs müssen bei einer Multi-LAB/2-Karte Basisadresse, PC-Interrupt- und PC-DMA-Kanal per Software eingestellt werden.

Falls zur Initialisierung nicht die mitgelieferten PC-Bibliotheken verwendet werden, so kann die Einstellung dieser Kartenparameter anhand der Erläuterungen dieser Application Note durchgeführt werden.

Weiterhin wird das Laden eines Betriebssystems auf eine Multi-LAB/2-Karte beschrieben. Dies muß ebenfalls vom PC aus gesteuert werden, da die Karte kein Boot-EPROM besitzt.

Zur vorliegenden Application Note steht ein Beispiel-C-Programm (BOOT2BSP.C) zur Verfügung, welches die Initialisierung der Multi-LAB/2-Karte und das Laden des Betriebssystems „OsX“ unter MS-DOS zeigt.

3. Der Plug-and-Play-Mechanismus der Multi-LAB/2-Karte

Die Multi-LAB/2-Karte besitzt ein jumperfreies Design, d.h. die Basisadresse kann per Software vom PC aus eingestellt werden. Der Plug-and-Play-Mechanismus bedient sich dafür der PC-I/O-Adresse 037bh.

Durch das Schreiben auf diese Adresse können verschiedene Aktionen ausgelöst werden, wobei die einzelnen Bits folgende Bedeutung haben:

PC-I/O-Adresse	Zugriff	Erklärung
037bh	W8	Plug-and-Play-Steuerregister der Multi-Lab/2-Karte: Bit 0 bis 2: Kartennummer (0..7), für die Aktion entsprechend Bit 6 und Bit 7 Bit 3 bis 5: reserviert, =0 setzen Bit 6 Bit 7: Aktion 0 0 Mit dem nächsten Schreibzugriff auf 037bh wird die Basis-Adresse gesetzt 1 0 Das Konfigurationsbyte (=01h) wird gesetzt, und zwar auf der Adresse, die mit dem nächsten Schreib-Zugriff auf 037bh gesetzt wird. x 1 Die aktuelle Basisadresse wird gelöscht. Danach ist die Karte nur noch über die PC-I/O-Adresse 037bh ansprechbar.

In der Funktion „SetMl2Adr“ (Abschnitt 6.2.) wird die Einrichtung einer Multi-LAB/2-Karte mit Kartennummer 0 unter der Basisadresse 02b8h gezeigt.

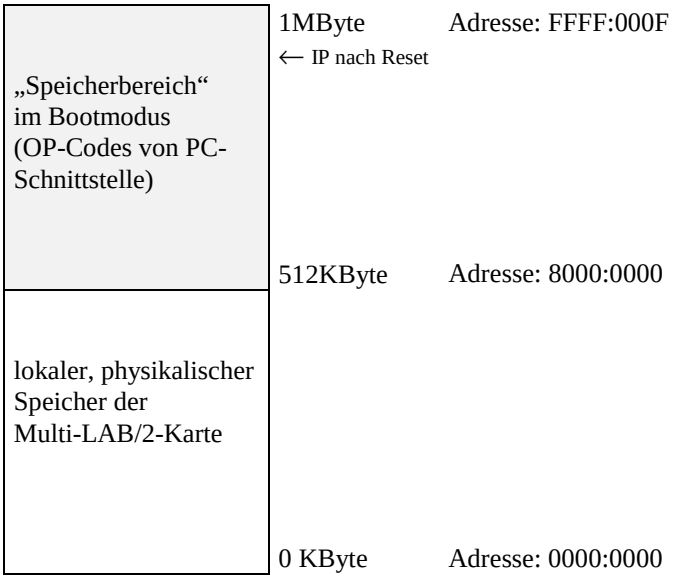
Nach der Einstellung der Basisadresse können PC-Interrupt- und PC-DMA-Kanal der Karte durch Schreiben auf die lokalen I/O-Adressen 84h bzw. 85h eingestellt werden. Die Bedeutung der einzelnen Bits sind aus dem Benutzer-Handbuch ersichtlich.

4. Die Steuerung des lokalen Prozessors

Da die Multi-LAB/2-Karte über kein lokales EPROM verfügt, aus dem ein Betriebssystem geladen und gestartet werden kann, muß der PC diese Aufgabe übernehmen. Die Befehlscodes (OP-Codes), die der lokale Prozessor üblicherweise aus dem lokalen Speicher lädt, werden im sog. Bootmodus über die PC-Schnittstelle übertragen.

Der 80x86-kompatible lokale Prozessor (V20) arbeitet im Real-Modus und kann 1MByte Speicher adressieren. Zeigt dessen IP auf die obere Hälfte dieses Speicherbereichs, so werden die Befehlscodes (OP-Codes) nicht aus dem lokalen Speicher, sondern von der PC-Schnittstelle der Multi-LAB/2-Karte abgeholt.

Nach einem Reset befindet sich die Multi-LAB/2-Karte im Bootmodus. Der Befehlszeiger (IP) des lokalen Prozessors zeigt auf eine Stelle am oberen Ende des Boot-Speicherbereichs, wie folgende Abbildung veranschaulicht:



Dem Statusregister der Multi-LAB/2-Karte kann entnommen werden, ob sich die Karte im Bootmodus befindet. Ist Bit 5=0, so wird die Karte im Bootmodus betrieben, ist Bit 5=1, so befindet sich die Karte im Standardmodus.

5. Vorgehensweise beim Laden eines Betriebssystems

Wird im folgenden auf Funktionen verwiesen, so sind deren Quell-Codes in Abschnitt 6.2. abgedruckt.

5.1 Prüfen des Zugriffsmodus

Es sind Fälle von PCs bekannt, bei denen aufgrund ungenauen AT-Bus-Timings nur 8-statt 16-Bit-Zugriffe verwendet werden können.

Daher sollte zu Beginn eine Überprüfung stattfinden, ob die Karte im 16-Bit-Modus betrieben werden kann.

In der Funktion "CheckBootMode" wird versucht, über einen 16-Bit-Zugriff das DLP-Bit im Statusregister der Karte zu setzen. Schlägt dies fehl, so müssen 8-Bit-Zugriffe verwendet werden, d.h. alle in den Listings über 16-Bit-Portbefehle ausgegebenen Wörter müssen in je zwei getrennt zu übertragende Bytes zerlegt werden.

5.2 Rücksetzen des Befehlszeigers

Bei jeder Ausführung eines OP-Codes durch den lokalen Prozessor wird dessen IP inkrementiert. Es ist zu beachten, daß die Multi-LAB/2-Karte bei einem Überlauf des IP von [FFFF:000F] nach [0000:0000] vom PC aus nicht mehr über OP-Codes ansprechbar ist, da sie sich dann OP-Codes aus dem lokalen Speicher lädt.

Da nach einem Reset der IP kurz vor dem Überlauf steht, sollte zunächst die Funktion „ResetCardIP“ aufgerufen werden.

5.3 Setzen des Datenzeigers und Schreiben von Daten

Um ein Betriebssystem auf der Karte starten zu können, muß zunächst die Betriebssystemdatei auf die Karte übertragen werden. Dazu wird in den Prozeduren „BootWriteRam“ und „SetSegOffsPointer“ gezeigt, wie der lokale Speicher der Multi-LAB/2-Karte beschrieben wird:

In einer Schleife wird zunächst der Datenzeiger des Prozessors gesetzt, der auf die zu beschreibende Stelle im lokalen Speicher zeigt, und dann ein Datenwort an diese Stelle geschrieben. So wird die gesamte Datei wortweise auf die Karte geladen.

5.4 Starten eines Programms

Nachdem sich das Betriebssystem im lokalen Speicher der Multi-LAB/2-Karte befindet, kann es durch Setzen des IP auf die Startadresse des Betriebssystems gestartet werden.

Dazu wird ein Far-Jump-Befehl als OP-Code an die Karte gesendet, wie es in der Funktion „BootStartPgm“ demonstriert wird.

5.5 Laden und Starten des Betriebssystems OsX

Die PDT des Betriebssystems OsX enthält alle zum Laden relevanten Parameter. Sie befindet sich am Anfang der Betriebssystemdatei. Ihre Struktur ist im Benutzer-Handbuch erläutert.

Zum Laden des OsX werden die Startadresse des Betriebssystems, die PDT-Länge und die Anzahl der Prozeduren, die das Betriebssystem enthält, benötigt. Diese Parameter werden zunächst aus der PDT gelesen. Aus diesen berechnet sich die Ladeadresse wie folgt:

(Ladeadresse) = (Startadresse) - 4*(Prozeduranzahl) - (PDT-Länge).

Ab der Ladeadresse wird nach Abschnitt 5.3 die gesamte Betriebssystemdatei in den lokalen Speicher geschrieben und danach, wie unter Abschnitt 5.4 beschrieben, gestartet.

Es ist wichtig, daß beim Starten des Betriebssystems PC und Multi-LAB/2 synchron arbeiten. Dazu sind die PC-Interrupts zu sperren. Nach dem Jump-Befehl an die Startadresse von OsX muß vom PC sofort ein Makrobefehl an die Karte gesendet werden, der keine Antwort erwartet, da die Multi-LAB/2-Karte nach dem Starten ihres Betriebssystems auf die PC-Schnittstelle zugreift. Erst wenn dieser Makrobefehl ausgeführt wurde (TBF=0), ist die Karte betriebsbereit. Der Synchronisationsmechanismus ist im Beispielprogramm „BOOT2BSP.C“ implementiert.

6. Listings

6.1 Allgemeine Bemerkungen zu den Listings

Der lokale Prozessor besitzt eine interne OP-Code-Pipeline. Damit ein OP-Code ausgeführt wird, muß mindestens ein weiterer folgen. Nach allen Befehlssequenzen der Beispiele wird daher mindestens ein NOP übertragen.

Werden OP-Codes wortweise an die Karte gesendet, so wird zunächst das Low-Byte, danach das High-Byte ausgeführt. Die Beschreibung der einzelnen OP-Codes in den Listings entspricht der Reihenfolge der Ausführung, also zuerst Low-Byte und dann High-Byte.

Die einzelnen Worte dürfen erst dann an die Karte gesendet werden, wenn die Schnittstelle frei ist. Dies garantiert die Prozedur „SendWord“. „SendByte“ besitzt denselben Aufbau wie „SendWord“ mit dem Unterschied, daß nur ein Byte gesendet wird.

Folgende Typendeklarationen werden in den Listings verwendet:

„ushort“ als „unsigned int“;

„byte“ als „unsigned char“.

6.2 Quellcode

Die folgenden Listings sind in dem Beispielprogramm „BOOT2BSP.C“ verarbeitet. Sie sind bewußt allgemein gehalten. Das konkrete Laden von OsX wird in „BOOT2BSP.C“ vorgestellt.

In den Funktionen werden folgende statische Variablen verwendet:

```
   /** Statische Variablen: **/
   static ushort stat_basisadr = 0x2B8;  /* Basisadresse der Karte */
   static ushort stat_RAMptr_segment;    /* aktueller RAM-Pointer */
   static ushort stat_RAMptr_offset;
```

Funktionen:

```
int SetMulti_LAB/2Adr(void)
    /*  Funktion: Setzen der Basisadresse einer Multi-LAB/2
        Rückgabe: 1: Fehler
                  0: gelungene Adresseinstellung
    */
{
  ushort cardnumber = 0;             /* Kartennummer; muß zwischen 0 und
                                     7 liegen*/
  ushort pc_io_adr = stat_basisadr;  /* Basisadresse, die eingestellt
                                     werden soll */
  ushort nba;                        /* A3 bis A10 der Basisadresse */
  byte formatbyte;                   /* Formatbyte zur Überprüfung der
                                     Plug&Play-Aktion */

  outp (0x37B, cardnumber | 0x80);   /* I/O-Adresse der Karte
                                     zurücksetzen. */
  outp (0x37B, cardnumber | 0x40);   /* Formatbyte-Adresse der
                                     angewählten Karte setzen. */
  nba = (pc_io_adr >> 2) & 0xFF;     /* es werden nur die Adress-Bits A3
                                     bis A10 übergeben. */
  outp (0x37B, nba);
  formatbyte = inp(pc_io_adr);       /* jetzt das Formatbyte lesen */
  outp (0x37B, cardnumber | 0x80);   /* I/O-Adresse wieder abschalten */

  if( formatbyte != 1 ) return(1);   /* Einstellung der Basisadresse
                                     fehlgeschlagen */

  outp( 0x37B, cardnumber );
  outp( 0x37B, nba );                /* die endgültige Basisadresse
                                     einstellen. */

/**   falls notwendig, kann ein Reset ausgelöst werden:   **/
  outp( pc_io_adr+3, 0x00 );         /* RESET der Karte auslösen */

  return(0);
}
```

```
ushort SendWord( ushort data )
    /* Funktion : Senden eines Worten an eine Multi-LAB/2
       Parameter:  data: zu sendendes Wort
       Rückgabe : 0: Senden hat geklappt
                  1: Timeout aufgetreten
    */
{
  byte status;                        /* Statuswert der Karte */
  ushort timeout;                     /* Timeout-Zähler */

#define TBF 1                         /* TBF Flag im Statusport */

  timeout = 20000U;                   /* Timeout-Zeit setzen */
  status  = inp( stat_basisadr );     /* Status der Karte lesen */
  while( ((status&TBF) != 0 ) &&      /* Ist das TBF-Flag gesetzt ?*/
         (timeout != 0 ) ) {          /* Timeoutzeit nicht abgelaufen */
     timeout--;
     status  = inp( stat_basisadr );/* Status der Karte lesen */
  }

  if( timeout == 0 )                  /* Wenn die Timeoutzeit = 0 ist,*/
     return( 1 );                     /*dann ist TBF-Fehler aufgetreten*/

  outpw( stat_basisadr+2, data );     /* Wort schreiben*/
  return( 0 );                        /* Kein Fehler */
}
```

```
int CheckBootMode( void )
    /* Funktion:  Test, ob 16-bit-Modus möglich ist
       Rückgabe:  1: 16-bit-Modus
                  0: 8-bit-Modus
                 -1: Karte nicht ansprechbar
   */
{
  byte status;                        /* Statuswert der Karte */

  SendWord( 0x87E6 );                 /* OUT 87h, al (DLP=1 setzen) */
  SendWord( 0x9090 );                 /* 2 NOPs */
  status  = inp( stat_basisadr );     /* Status der Karte lesen */
  if( (status&0x40) == 0x40 ) {       /* Falls DLP=1 gesetzt */
      SendWord( 0x86E6 );             /* OUT 86h, al (DLP=0 setzen) */
      SendWord( 0x9090 );             /* 2 NOPs */
      return(1);                      /* 16-bit-Modus setzen */
  } else {
      SendByte( 0xE6 );               /* OUT 87h, al (DLP=1 setzen) */
      SendByte( 0x87 );
      SendByte( 0x90 );               /* NOP */
      status  = inp( stat_basisadr );/* Status der Karte lesen */
      if( (status&0x40) == 0x40 ) {  /* Falls DLP=1 gesetzt */
         SendByte( 0xE6 );            /* OUT 86h, al (DLP=0 setzen) */
         SendByte( 0x86 );
         SendByte( 0x90 );            /* NOP */
         return(0);                   /* 8 Bit Mode setzen */
      }
  }
  return(-1);                         /* DLP setzen fehlgeschlagen */
}
```

```
ushort ResetCardIP( void )
    /* Funktion: Setzen des IP auf 8000:0000, d.h. auf den Anfang des
                 Boot-Speicherbereichs
       Rückgabe: 0: IP zurückgesetzt
                >0: Fehler aufgetreten
    */
{
  ushort error;                      /* Fehlervariable */

  error = 0;
  error += SendWord( 0x00EA );       /* EA: Long-Jump; Offset low-Byte*/
  error += SendWord( 0x0000 );       /*Offset high-Byte; Seg. low-Byte*/
  error += SendWord( 0x9080 );       /*Segment high-Byte; NOP*/
  return( error );
}
```

```
ushort SetSegOffPointer( ushort adr_seg, ushort adr_off )
    /* Funktion : Setzen des RAM-Zeigers einer Multi-LAB/2
       Parameter: adr_seg: Segmentadresse
                  adr_off: Offsetadresse
       Rückgabe : 0: RAM-Adresszeiger gesetzt
                 >0: Fehler aufgetreten
    */
{
  byte lo_adr_seg, hi_adr_seg;       /* Segment low- und high-Byte*/
  byte lo_adr_off, hi_adr_off;       /* Offset low- und high-Byte */
  ushort error;                      /* Fehlervariable */

  error = 0;
                                     /* Aufteilen von Seg.und Offset */
  lo_adr_seg = adr_seg & 0xFF;       /* in Low- und High-Byte */
  hi_adr_seg = (adr_seg >> 8) & 0xFF;
  lo_adr_off = adr_off & 0xFF;
  hi_adr_off = (adr_off >> 8) & 0xFF;

  error += SendWord( 0xB890 );       /* NOP; MOV AX,Segment */
  error += SendWord( adr_seg );      /* [Segment] */
  error += SendWord( 0xD88E );       /* MOV DS,AX */
  error += SendWord( (((ushort) lo_adr_off) << 8) + 0xBB );
                                     /* MOV BX,Offset; Offset low */
  error += SendWord( hi_adr_off + 0x9000 ); /* Offset high; NOP*/
  return( error );
}
```

```
ushort BootWriteRam( ushort size, byte *data_var )
    /* Funktion : schreibt einen Datenblock in das lokale RAM
       Parameter: size    : Größe des Datenbereichs
                  data_var: Zeiger auf den Datenbereich
       Rückgabe:  0: fehlerfreie Ausführung
                 >0: Fehler aufgetreten
    */
{
  int i;                            /* Schleifenzähler */
  byte data;                        /* Aktuelles Datenbyte */
  ushort error;                     /* Fehlervariable */
  ushort adr_seg;                   /* RAM-Zeiger, Laufvariablen */
  ushort adr_off;

  error = 0;

  adr_seg = stat_RAMptr_segment;    /* Laufvariablen initialisieren */
  adr_off = stat_RAMptr_offset;

  for(i=0;i<size;i++) {
     data = *data_var;
     data_var++;
     SetSegOffPointer( adr_seg, adr_off );
                                    /* Neue Adresse setzen */
     error += SendWord( 0x07C6 );   /* MOV BYTE PTR [BX],Data */
     error += SendWord( data + 0x9000 );
                                    /* [data] ; NOP*/
     adr_off++;                     /* Adresse erhöhen */
     if( adr_off == 0 )             /* Überlauf ? */
         adr_seg += 0x1000;         /* Segment erhöhen */
  }
  stat_RAMptr_segment = adr_seg;    /* statische Variablen */
  stat_RAMptr_offset = adr_off;     /* aktualisieren */

  return( error );
}
```

```
ushort BootStartPgm( void )
    /* Funktion : Start eines Programms auf der Multi-LAB/2
       Rückgabe:  0: fehlerfreie Ausführung
                 >0: Fehler aufgetreten
    */
{
  byte lo_adr_seg, hi_adr_seg;
  byte lo_adr_off, hi_adr_off;
  ushort error;                     /* Fehlervariable */

  error = 0;
                                    /* Aufteilen von Seg. und Offset */
                                    /* in Low- und High-Byte */
  lo_adr_seg = stat_RAMptr_segment & 0xFF;
  hi_adr_seg = (stat_RAMptr_segment >> 8) & 0xFF;
  lo_adr_off = stat_RAMptr_offset & 0xFF;
  hi_adr_off = (stat_RAMptr_offset >> 8) & 0xFF;

  error += SendWord( (((ushort) lo_adr_off) << 8 ) + 0xEA );
                                    /* EA: Long Jump; Offset low */
  error += SendWord( (((ushort) lo_adr_seg) << 8 ) + hi_adr_off );
                                    /* Offset high; Segment low */
  error += SendWord( 0x9000 + hi_adr_seg );
                                    /* Segment high; NOP */
  return( error );
}
```