SYSTECH J.Schnyder GmbH

Schliefweg 30
CH-4106 Therwil
Telefon 091 827 15 87
www.systech-gmbh.ch

Tutorial
EBS08-GB/AB

Steppermotor Controller

Version for Metrowerks Codewarrier 5.0

Version 0.3a (Draft)

Part 1
Basic System,
Keyboard and Keyboard Interpreter

Contents

Part 1 1
- Preface 3
- How the software will work 3
- First steps 4
  - a) Preparing 4
- Including the software module for the keyboard 4
  - a) the module itself 4
  - b) the variables and declarations needed 5
  - c) Inserting the module in the task manager 6
  - d) How it works 8
- Creating a keyboard interpreter 8
  - a) Creating a new file 8
  - b) Including and initializing the new file 9
  - c) The keyboard interpreter 11
  - d) The code for the keyboard interpreter (first part) 11

Part 2 15
- Appendix A Ports 16
- Appendix B Keyboard 18
  - Component placing 18
- Links 20

Preface

This tutorial intends to explain how to use the EBS08 operating system.
It also has the scope to show how to use the following modules:
- 4x4 keyboard MODULE
- math functions
- LCD driver
- stepper motor module.

further it gives a idea how stepper motors are working.

How the software will work

The hardware consists of a 4x4 keyboard n port F, 6 LEDs on port E (bits2..), a LCD module on port D and a stepper motor interface on port B.

On the keyboard the key are used like the following

- ..... 100 Steps CCW	- ..... 10 Steps CCW	- ..... 10 Steps CW	- ..... 100 Steps CW
- ..... Fast CCW	- ..... Slow CCW	- ..... Slow CW	- ..... Fast CW
Full steps 1 Phase ..... -	Full steps 2 Phase ..... -	Half steps ..... -	Boost off ..... Boost on
SHIFT			

After a reset the controller is in half step mode, the stepper motor is stopped and the boost function is off.

First steps

a) Preparing

Read the Tutorial for creating a new EBS08 project with CW 5.0 (this can be downloaded from http://systech-gmbh.ch -> EBS08 -> Tutorials and prepare a new project named:

"**EBS08_SM**"

Now we are ready for the next steps!

(This level is stored as ebs08_sm_tut_ab_step1.zip or ebs08_sm_tut_ab_step1.zip; the whole project!)

We will use xx for either GB or AB in the following sections. So both microcontrollers can use the same tutorial.

Including the software module for the keyboard

Download the **tast4x4.zip** file then unzip it. The unziped files will be copied into the "c:\hc08\projects\allgemein" directory. Then copy **tast4x4.mmw** in the **source** directory of the **EBS_SM** project, add the **tast4x4.mmw** file with a right click on the **source** directory to the project

a) the module itself

Open the file main.asm in the metrowerks project window by double clicking on it. Then look for the following text:

```
    ORG  ROM
main:
_Startup:

  include "xx_ini.mmw"            ;Initialisierung
  include "xx_tsk.mmw"            ;Task-Manager


  include "xx_tic.mmw"            ;TIC Erzeugung

  include "i_dummy.mmw"           ;Dummy Interrupt

    ORG  VECTOR
  include "xx_vec.mmw"            ;Interrupt-Vektoren
```

To include the module insert the following marked in bold:

```
    ORG  ROM
main:
_Startup:

  include "xx_ini.mmw"            ;Initialisierung
  include "xx_tsk.mmw"            ;Task-Manager
```

```
  include "tast4x4.mmw"            ;Keyboard module

  include "xx_tic.mmw"             ;TIC Erzeugung

  include "i_dummy.mmw"            ;Dummy Interrupt

    ORG VECTOR
  include "xx_vec.mmw"             ;Interrupt-Vektoren
```

b) the variables and declarations needed

Look for the following in the main file:

```
* Systemtakt 8.000 MHZ
* Prozessor-Zyklus Tp = 250 (1/f * 2)   BUS-Clock

* 1 ms = 4000  Zyklen

TIC      EQU     4000
TICH     EQU     TIC / 256             ;high = TIC/256
TICL     EQU     TIC-TICH*256          ;low = TIC-TICH*256

WDOG     EQU     SRS                   ;WDOG Zuweisung

* Speicherbelegung
```

the keyboard module needs the following declarations (see also: tast4x4.def):

The ports:

TST_PRT	the port where the keyboard will be attached
TST_DDR	the datadirection register of this port
TST_PUE	the pullup register for this port

The variables:

TST_F	the Flags for this sub-pgm
TST_Z	the counter for the scans
TST_ZP	the debouncing counter
TST_NR	the counter for the keys detected
TST_P0	the first key
TST_P1	the second key

so we add:

```
* Systemtakt 8.000 MHZ
* Prozessor-Zyklus Tp = 250 (1/f * 2)   BUS-Clock

* 1 ms = 4000  Zyklen

TIC      EQU     4000
TICH     EQU     TIC / 256             ;high = TIC/256
TICL     EQU     TIC-TICH*256          ;low = TIC-TICH*256

WDOG     EQU     SRS                   ;WDOG Zuweisung
```

```
* Ports

* Keyboard                                  ;see also APPENDIX A
for GB micros write:
TST_PRT  EQU      PTAD                     ;Port A
TST_DDR  EQU      PTADD                    ;DDR Port A
TST_PUE  EQU      PTAPE                    ;PUE Port A

for AB micros write:
TST_PRT  EQU      PORTF                    ;Port F
TST_DDR  EQU      DDRF                     ;DDR Port F
TST_PUE  EQU      PTDPUF                   ;PUE Port F

* Speicherbelegung
```

Now we have to declare the variables.
It should look like the following:

```
Z_CORE   RMB      1         ;Core-Zähler in 1TIC Schritten

TASK_F1  RMB      1         ;Task-Manager Flags1 1=

TST_F    RMB      1         ;Flags for the keyboard
                            ;see in tast4x4
TST_Z    RMB      1         ;Counter for the scans
TST_ZP   RMB      1         ;Storage for debouncing
TST_NR   RMB      1         ;Counter for the keys and result
TST_P0   RMB      1         ;First key
TST_P1   RMB      1         ;Second key

* temporäre Speicher

MAT0     RMB      1         ;Hilfsvariabele für Math-Funktionen
MAT1     RMB      1
```

c) Inserting the module in the task manager

Open the module xx_tsk.mmw the so called task manager by double clicking on it.

Here you see the code:

```
* Erstellt:      06.06.2005 st-js
* Ergänzt:


T_S:     BRCLR    F_TIC1,CORE_F,T_S       ;warten auf Start einer
                                           1ms-Einheit
         BCLR     F_TIC1,CORE_F           ;Flag löschen

         STA      WDOG                    ;Watchdog zurücksetzen

T_SX:
```

```
* hier wird entschieden was während diesem TIC getan werden soll
* here you have to call your tasks to do!


  LDA   Z_CORE
  STA   PTFD

T_E:
        JMP     T_S                     ;und von vorne
```

First delete the two lines shown below : (they should look similar, we do not need any example, so delete all between ..... **to do!** and the label **T_E:**

```
  LDA   Z_CORE
  STA   PTFD
```

then insert the following (marked in bold):

```
* Erstellt:     06.06.2005 st-js
* Ergänzt:


T_S:    BRCLR   F_TIC1,CORE_F,T_S       ;warten auf Start einer
                                         1ms-Einheit
        BCLR    F_TIC1,CORE_F           ;Flag löschen

        STA     WDOG                    ;Watchdog zurücksetzen

T_SX:

* hier wird entschieden was während diesem TIC getan werden soll
* here you have to call your tasks to do!

T_KEY:
    BRCLR    F_TIC8,CORE_F,T_KEYE      ;every 8TICs (8ms)
    JSR      TAST                      ;call the keyboard routine
T_KEYE:

T_I:
    BRCLR    F_ZOK,TST_F,T_KBDIE       ;if there is a key pressed
                                       ;we show the result on port F
    LDA      TST_NR                    ;get the result
    for GB micros write:
    STA      PTFE                      ;and put it on port F
    for AB micros write:
    STA      PORTE                     ;and put it on port E

T_KBDIE:

T_E:
        JMP     T_S                     ;und von vorne
```

We have now to initialize the ports and variables of the tast4x4 module.
Open the module xx_ini-mmw and insert the following:

```
INI_PRT:
        LDA     #%00000000
        STA     PTAD                    ;Port A Input!
        LDA     #%00000000
        STA     PTADD
```

```
        LDA     #%11111111              ;Port B alles OUTPUT
        STA     PTBDD

        STA     TST_DDR                   ;Keyboard-Port -> Output

INI_TAST:
        JSR     TAST_INI                ;init. of the keyboard module

INI_VAR:
        CLRA
        STA     TASK_F1                 ;Task-Manager Variable


INI_INT:
        JSR     START_TIC               ;TIC starten
```

Now we are ready to test the first part of our program. Connect the keyboard on port A (Port F for AB micros) and connect the target via BDM with your PC.

Make sure you have selected the right target (i.e. SofTec inDart). Then click on the green Bug (Debug) and the program will be compiled, and if no errors had occurred, the code will be downloaded on the board.

If you press a key the code of this key will be displayed on port F. Be sure to have removed the jumpers on the demoboard (if there are any)! If you press two keys at the same time their code will be displayed in the low and hi nibble of port F.

You will not see keycode $00 (because with $00 no LED will be ON).

d) How it works

First the module tast4x4 is initialized by the subroutine TAST_INI.

The line you have inserted in the task manager calls every 8TICs (in our case every 8ms) the subroutine TAST. Every call of TAST scans one column of the keyboard matrix by reading all the four row bits when the selected column is driver low. The fifth call of TAST then calculates the keycode (if a key was pressed) and stores it in the debouncing buffer TST_ZP. If the subroutine TAST detects two more times (10 scans later) the same keycode, this code is valid and will be moved in TST_NR and the F_ZOK Flag in the TST_F variable so that the task manager can react on this event. In our case (for testing purposes) the task manages simply writes the keycode to port F.

(This level is stored as ebs08_sm_tut_ab_step2.zip or ebs08_sm_tut_gb_step2.zip; source files only!)

Creating a keyboard interpreter

a) Creating a new file

From the menu **File** select **New Text File**. A window will with a file named **untitled** will be created and we can write the following:

```
* sub.mmw       subpgm for the sm project

* created:      06.03.2006    st-js
* modified:

SUB_KBDI:          ;command interpreter

SUB_KBDIE:
        RTS
```

Save the file. (Menu **File** -> **Save**). Place the file in the source directory of the project named "**sm_sub.mmw**". This will be the base for our keyboard interpreter.

Now we have to include this file in our project first in the MW environment by right clicking on **sources** and choosing **Add Files**. In ...\sources we select **sm_sub.mmw** then **Open**.

b) Including and initializing the new file

Open main.asm and insert the following:

```
main:
_Startup:

  include "xx_ini.mmw"            ;Initialisierung
  include "xx_tsk.mmw"            ;Task-Manager

  include "tast4x4.mmw"           ;Keyboard module
  include "sm_sub.mmw"            ;Subroutines

  include "xx_tic.mmw"            ;TIC Erzeugung

  include "i_dummy.mmw"           ;Dummy Interrupt

    ORG VECTOR
  include "xx_vec.mmw"            ;Interrupt-Vektoren
```

we have also to define the LED-Port as below in the main.asm file:

```
WDOG    EQU     SRS                 ;WDOG Zuweisung

* Ports

* Keyboard
TST_PRT  EQU  PTAD                 ;Port A
TST_DDR  EQU  PTADD                ;DDR Port A
TST_PUE  EQU  PTAPE                ;PUE Port A

* LED_Port

for the GB micro write:
LED_PRT EQU     PTFD               ;Port F
LED_DDR EQU     PTFDD              ;DDR Port F

for the AB micro write:
LED_PRT EQU     PORTE              ;Port E
LED_DDR EQU     DDRE               ;DDR Port E

for both of them:
```

```
L_FULL  EQU     2                       ;FULLSTEP MODE
L_2PH   EQU     3                       ;2 PHASE MODE
L_BOOST EQU     7                       ;BOOST ON

* Speicherbelegung
```

Change in the xx_ini.mmw file:

```
INI_PRT:
        LDA     #%00000000
        STA     PTAD (PORTA)        ;Port A Input!
        LDA     #%00000000
        STA     PTADD (DDRA)

        LDA     #%11111111          ;Port B alles OUTPUT
        STA     PTBDD (DDRB)

        STA     LED_DDR             ;LED-Port for Output
```

In the task manager (xx_tsk.mmw) we change the output of the keycode on port F (Port E for the AB micro) in a call of the command interpreter in case there is a keycode present.

```
T_SX:

* hier wird entschieden was während diesem TIC getan werden soll
* here you have to call your tasks to do!

T_KEY:
    BRCLR   F_TIC4,CORE_F,T_KEYE        ;every 8TICs (8ms)
    BRSET   F_ZOK,TST_F,T_KEYE          ;execute TAST only if the command
                                        ;interpreter has checked the
                                        ;keycode

        JSR  TAST                       ;call the keyboard routine
T_KEYE:

T_KBDI:
        BRCLR   F_ZOK,TST_F,T_KDIE;if there is a key pressed

    JSR     SUB_KBDI                    ;comand interpreter
    BCLR    F_ZOK,TST_F                 ;clear the keycode OK flag
T_KBDIE:

T_E:
        JMP     T_S                     ;und von vorne
```

c) The keyboard interpreter

Now have to look at the keycodes

Name	Keycode	Action
100 Steps CCW	$00	100 steps counter clockwise
10 Steps CCW	$04	10 steps counter clockwise
10 Steps CW	$08	10 steps clockwise
100 Steps CW	$0C	100 steps clockwise
Fast CCW	$01	fast motion counter clockwise (until key release)
Slow CCW	$05	slow motion counter clockwise (until key release)
Slow CW	$09	slow motion clockwise (until key release
Fast CW	$0D	fast motion clockwise (until key release
SHIFT	$03	NONE
SHIFT + Full Steps 1 Phase	$32	sets controller in full step 1 phase mode
SHIFT + Full Steps 2 Phase	$63	sets controller in full step 2 phase mode
SHIFT + Half Steps	$A3	sets controller in half step mode
Boost on	$0E	boost the stepper motor current
SHIFT + Boost off	$E3	stepper motor current low

d) The code for the keyboard interpreter (first part)

So we have to decode 12 actions. You have to write the following code in the xx_sub.mmw file:

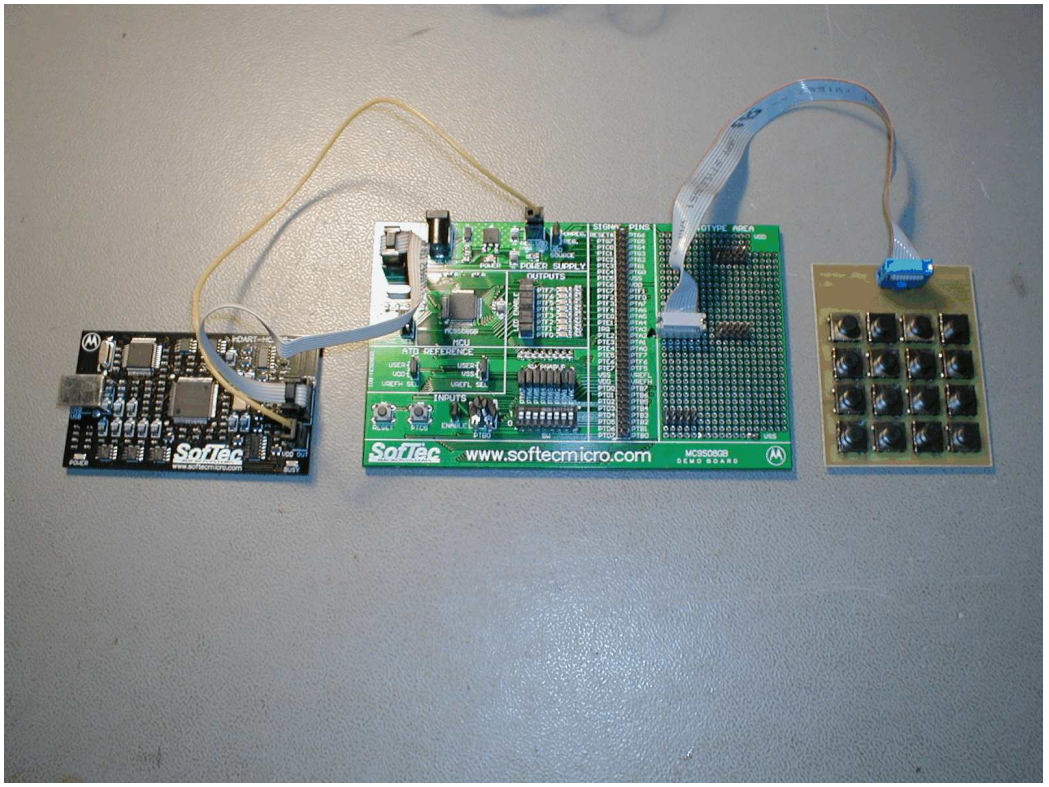
```
* sub.mmw        subpgm for the sm project

* created:       04.07.2005      st-js
* modified:      09.07.2005      st-js   renaming from CMDI in KBDI


SUB_KBDI:                                ;keyboard interpreter
        LDA     TST_NR                   ;get the keycode
SUB_KBD00:
        CMP     #$00
        BNE     SUB_KBD01
        JMP     SUB_K00                  ;Key $00
SUB_KBD01:
        CMP     #$01                     ;Key $01
        BNE     SUB_KBD02
        JMP     SUB_K01
SUB_KBD02:
                                         ;left for additional keycodes
```

```
SUB_KBD04
        CMP     #$04                    ;Key $04
        BNE     SUB_KBD05
        JMP     SUB_K04
SUB_KBD05:
        CMP     #$05                    ;Key $05
        BNE     SUB_KBD06
        JMP     SUB_K05
SUB_KBD06:

SUB_KBD08:
        CMP     #$08                    ;Key $08
        BNE     SUB_KBD09
        JMP     SUB_K08
SUB_KBD09:
        CMP     #$09                    ;Key $09
        BNE     SUB_KBD0A
        JMP     SUB_K09
SUB_KBD0A:

SUB_KBD0C:
        CMP     #$0C                    ;Key $0C
        BNE     SUB_KBD0D
        JMP     SUB_K0C
SUB_KBD0D:
        CMP     #$0D                    ;Key $0D
        BNE     SUB_KBD0E
        JMP     SUB_K0D
SUB_KBD0E:
        CMP     #$0E                    ;Key $0E
        BNE     SUB_KBD0F
        JMP     SUB_K0E
SUB_KBD0F:

SUB_KBD32:                              ;Key $02 and Key $03
        CMP     #$32
        BNE     SUB_KBD43
        JMP     SUB_K32
SUB_KBD43:

SUB_KBD63:                              ;Key $03 and Key $06
        CMP     #$63
        BNE     SUB_KBD73
        JMP     SUB_K63
SUB_KBD73:

SUB_KBDA3:                              ;Key $03 and Key $0A
        CMP     #$A3
        BNE     SUB_KBDB3
        JMP     SUB_KA3
SUB_KBDB3:

SUB_KBDE3:                              ;Key $03 and Key $0E
        CMP     #$E3
        BNE     SUB_KBDF3
        JMP     SUB_KE3
SUB_KBDF3:
        JMP     SUB_KBDIE               ;no keycode valid -> no action
```

```
SUB_K00:                                ;100 STEPs CCW

SUB_K01:                                ;FAST CCW

SUB_K04:                                ;10 STEPs CCW

SUB_K05:                                ;SLOW CCW

SUB_K08:                                ;10 STEPs CW

SUB_K09:                                ;SLOW CW

SUB_K0C:                                ;100 STEPs CW

SUB_K0D:                                ;FAST CW
        JMP     SUB_KBDIE

SUB_K0E:                                ;BOOST ON
        BSET    L_BOOST,LED_PRT         ;LED BOOST ON
        JMP     SUB_KBDIE

SUB_K32:                                ;FULLSTEP 1 PHASE
        BSET    L_FULL,LED_PRT          ;LED FULLSTEP ON
        BCLR    L_2PH,LED_PRT           ;LED 2PHASE OFF
        JMP     SUB_KBDIE

SUB_K63:                                ;FULLSTEP 2 PHASE
        BSET    L_FULL,LED_PRT          ;LED FULLSTEP ON
        BSET    L_2PH,LED_PRT           ;LED 2PHASE ON
        JMP     SUB_KBDIE

SUB_KA3:                                ;HALFSTEP
        BCLR    L_FULL,LED_PRT          ;LED FULLSTEP OFF
        BCLR    L_2PH,LED_PRT           ;LED 2PHASE OFF
        JMP     SUB_KBDIE

SUB_KE3:                                ;BOOST OFF
        BCLR    L_BOOST,LED_PRT         ;LED BOOST OFF
        JMP     SUB_KBDIE

SUB_KBDIE:
        RTS
```

Now we are ready to test the first part of this tutorial. Compile the whole program and download it on the evaluation board. With the keyboard attached you can test the functions of the mode keys be pressing **SHIFT** and the **Fullstep-1Phase**, **Fullstep-2Phase** and **Halfstep** keys. The LEDs will act as output for the mode selected now (in the second part we will add the moving functions of the stepper motor). If you select the **Boost** key or **SHIFT** and the **Boost** key the Boost LED will go on or off.

(This level is stored as ebs08_tut_ab_step3.zip or ebs08_tut_ab_step3.zip ; source files only!)



The keyboard interface attached to the HCS908GB60 evaluation board with the KBD

Part 2

Coming soon:
The stepper motor module and some math functions

Appendix A Ports

HCS08GBxx Ports

Pins for LQFP 64

Port A

PTA0 KBI1P0	46	ROW0
PTA1 KBI1P1	47	ROW1
PTA2 KBI1P2	48	ROW2
PTA3 KBI1P3	49	ROW3
PTA4 KBI1P4	50	COL0
PTA5 KBI1P5	51	COL1
PTA6 KBI1P6	52	COL2
PTA7 KBI1P7	53	COL3

Port E

PTE0 TxD1	14	
PTE1 RxD1	15	
PTE2 /SS1	17	
PTE3 MISO1	18	
PTE4 MOSi1	19	
PTE5 SPSCK1	20	
PTE6	21	
PTE7	22	

Port B

PTB0 AD1P0	33	
PTB1 AD1P1	34	
PTB2 AD1P2	35	
PTB3 AD1P3	36	
PTB4 AD1P4	37	
PTB5 AD1P5	38	
PTB6 AD1P6	39	
PTB7 AD1P7	40	

Port F

Pulldown konfigurierbar wenn Eingang

PTF0	54	
PTF1	55	
PTF2	11	LED fullstep
PTF3	12	LED 2 phase
PTF4	13	LED ...
PTF5	43	LED ...
PTF6	44	LED reverse
PTF7	45	LED boost

Stepper-Motor

Port C

PTC0 TxD2	3	PH1+
PTC1 RxD2	4	PH1-
PTC2 SDA1	5	PH2+
PTC3 SCL1	6	PH2-
PTC4	7	BOOST
PTC5	8	
PTC6	9	
PTC7	10	

KEYPAD-Port

Port G

PTG0 BKGD/MS	58	
PTG1 XTAL	59	
PTG2 EXTAL	60	
PTG3	61	
PTG4	62	
PTG5	63	
PTG6	64	
PTG7	2	

LCD-Port

Port D

PTD0 TPM1CH0	25	DB4
PTD1 TPM1CH1	26	DB5
PTD2 TPM1CH2	27	DB6
PTD3 TPM2CH0	28	DB7
PTD4 TPM2CH1	29	R/W
PTD5 TPM2CH2	30	RS
PTD6 TPM2CH3	31	Enable
PTD7 TPM2CH4	32	

IRQ	16	

HC08AB32 Belegung Tutorial

27/06/2005

Port A

PTA0 mon8!	26/A8	res.
PTA1	27/A9	
PTA2	28/A10	
PTA3	29/A11	
PTA4	30/A12	
PTA5	31/A13	
PTA6	32/A14	
PTA7	33/A15	

Port E

PTE0 TxD	13/C2	TX res.
PTE1 RxD	14/A2	RX res.
PTE2 TACH0	15/C3	LED fullstep
PTE3 TACH1	16/A3	LED 2 phase
PTE4 /SS	17/C4	LED ...
PTE5 MISO	18/A4	LED ...
PTE6 MOSI	19/C5	LED reverse
PTE7 SPSCK	20/A5	LED boost

A/D-Wandler

Port B

PTB0 ATD0	34/C8	
PTB1 ATD1	35/C9	
PTB2 ATD2	36/C10	
PTB3 ATD3	37/C11	
PTB4 ATD4	38/C12	
PTB5 ATD5	39/C13	
PTB6 ATD6	40/C14	
PTB7 ATD7	41/C15	

KEYPAD-Port

Port F

Pulldown konfigurierbar wenn Eingang

PTF0 TACH2	4/C24	ROW0
PTF1 TACH3	5/A24	ROW1
PTF2 TBCH2	6/C25	ROW2
PTF3 TBCH3	7/A25	ROW3
PTF4 TBCH0	8/C28	COL0
PTF5 TBCH1	11/A28	COL1
PTF6	12/C29	COL2
PTF7	10/A29	COL3

Stepper-Motor

Port C

PTC0 mon8!	60/A16	PH1+
PTC1 mon8!	61/A17	PH1-
PTC2 MCLK	62/A18	PH2+
PTC3 mon8!	63/A19	PH2-
PTC4	1/A20	BOOST
PTC5	64/A21	

Port G

PTG0 KBD0	23/C7
PTG1 KBD1	24/A7
PTG2 KBD2	25/C23!

LCD-Port

Port D

Hi-Current Ports

PTD0	42/C16	DB4
PTD1	43/C17	DB5
PTD2	46/C18	DB6
PTD3	47/C19	DB7
PTD4	50/C20	R/W
PTD5	51/C21	RS
PTD6 TACLK	52/C22	Enable
PTD7 TBCLK	53/C23!	

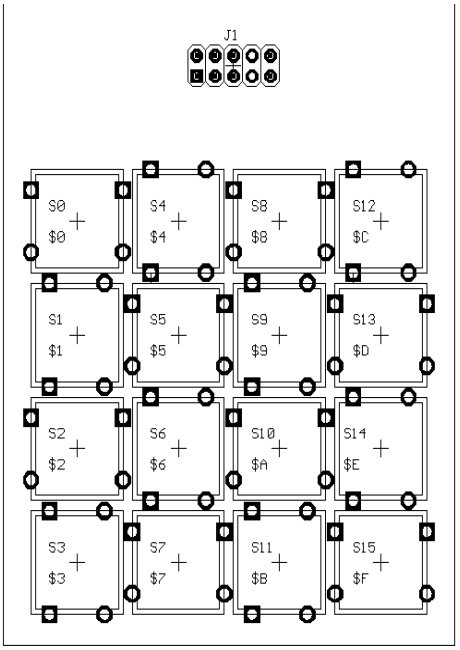
Port H

PTH0 KBD3	48/A22
PTH1 KBD4	49/A23

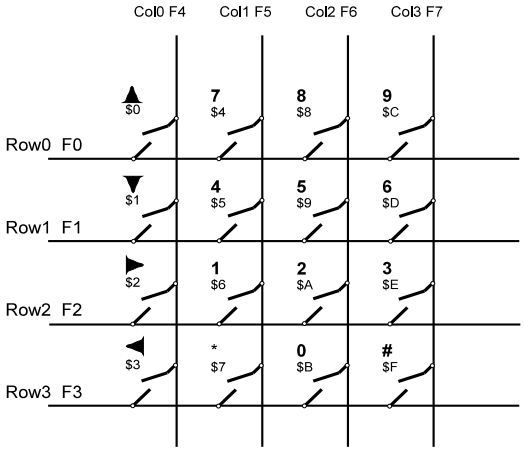
	/IRQ	2/C6

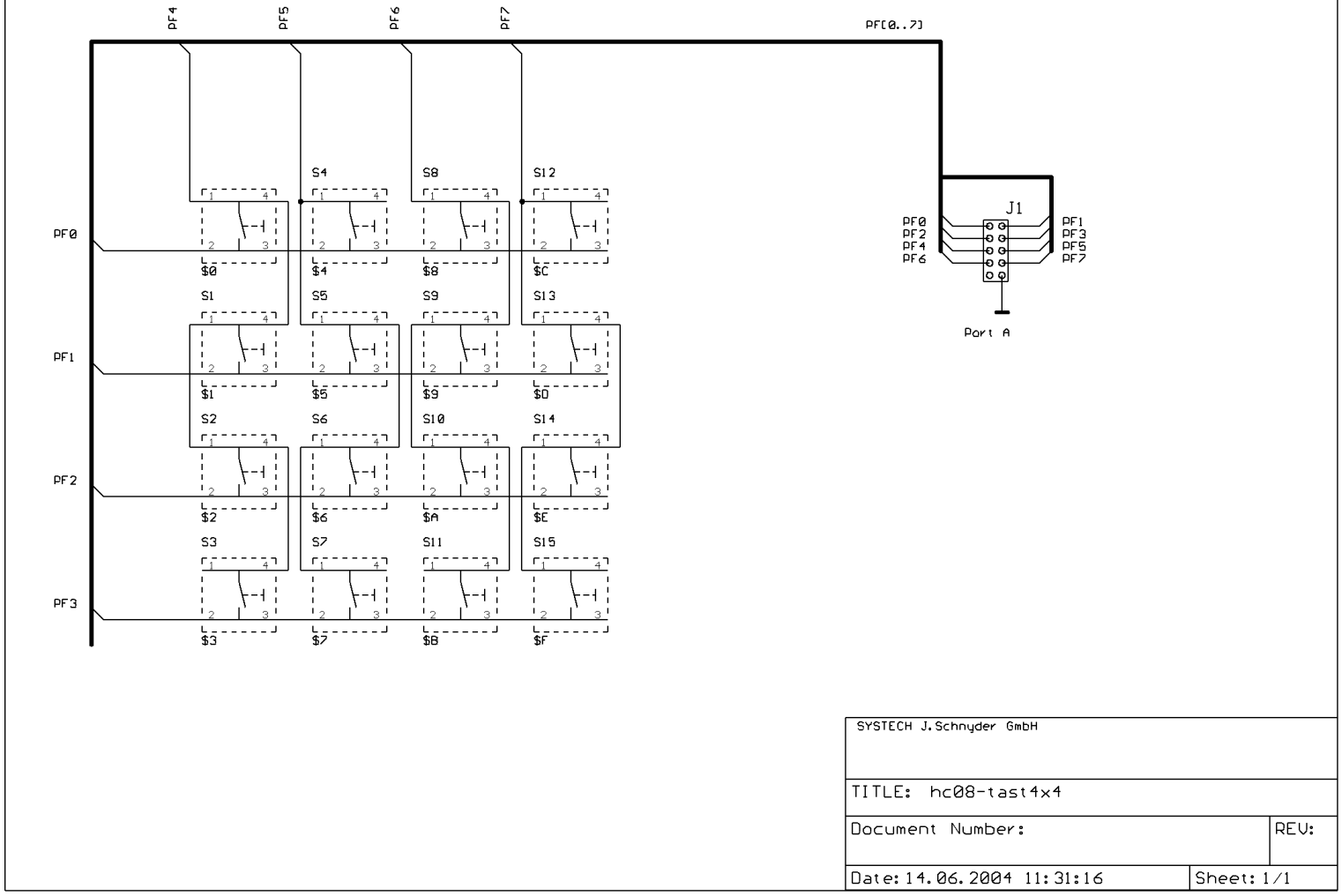
Appendix B Keyboard

Component placing



TAST4x4





SYSTECH J.Schnyder GmbH	
TITLE: hc08-tast4x4	
Document Number:	REV:
Date: 14.06.2004 11:31:16	Sheet: 1/1

Links

Distrelec www.distrelec.com

components und more

Systech J.Schnyder GmbH www.systech-gmbh.ch

development of hard- und software, educational systems
layout-programs and more

Freescale forums.freescale.com

software and aid for Freescale micros

ucgeeks www.ucgeeks.com

software and aid for micros