Application Note, V 2.0, Feb. 2004

AP16018

C167CS-32FM

TFT graphic display driven by C167CS-32FM

Microcontrollers

Never stop thinking.

Revision History:	2004-02	**V 2.0**
Previous Version:	V 1.0	
Page	Subjects (major changes since last revision)	
All	Updated layout to Infineon Corporate Design, updated revision to 2.0, Content unchanged!	

Controller Area Network (CAN): License of Robert Bosch GmbH

We Listen to Your Comments

Any information within this document that you feel is wrong, unclear or missing at all? Your feedback will help us to continuously improve the quality of this document. Please send your proposal (including a reference to this document) to:
mcdocu.comments@infineon.com

Table of Contents **Page**

1 General description

1.1 Picture format design flow

Realization of the Hardware and Software connectiong an C167CS-32FM starterkit to an TFT display with a screen diagonal of 7“ for e.g. dashboard / cluster / multifuncional terminal as an application example.

This application should show easily, how to realize the access to a TFT display by using a C167CS-32FM and 8 bit non mux bus with 3 waitstates and asynchronous ready via BUSCON 4.

A blinking oilcan is to be shown on the display as an alert announcement for the driver in order to check his ammount of oil regarding the combustion engine.

The oilcan will be designed in COREL DRAW or an another suitable grafic design tool.

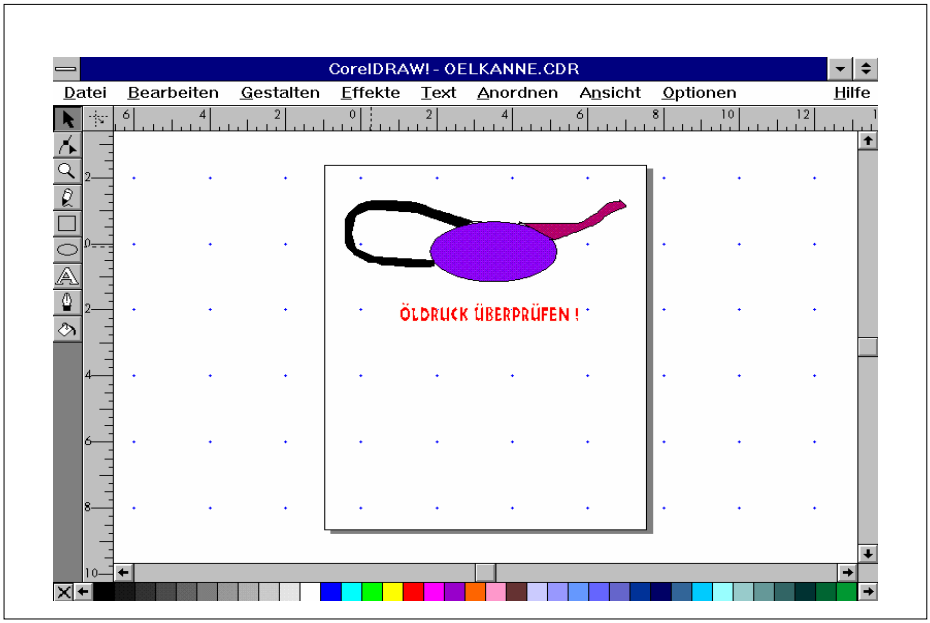

Figure 1 Oilcan in Corel Draw

Adapted to the pixel format with PAINT SHOP PRO 6 according the size of the used display.

Figure 2 Changing the pixel format with Paint Shop Pro

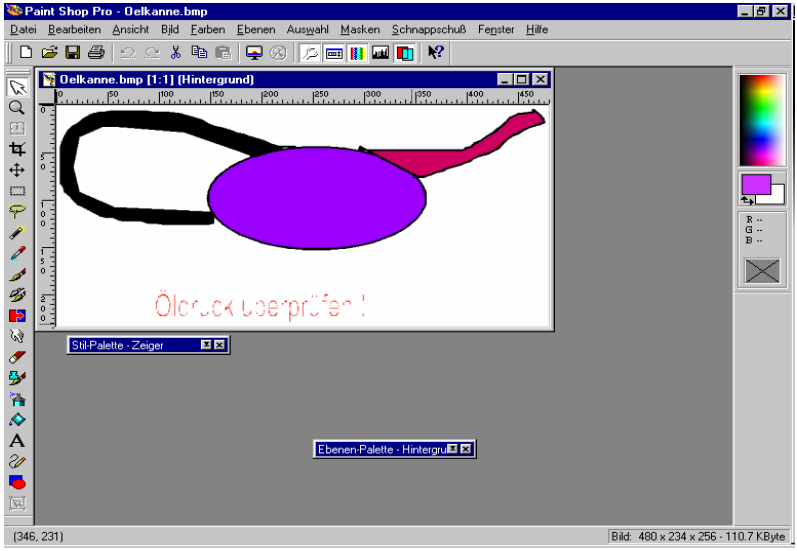

Figure 3 Transformed into a WIN / OS bit map format

Formated finally with ULTRA EDIT into a readable array in a HEADER file for KEIL or TASKING C166 Compiler.

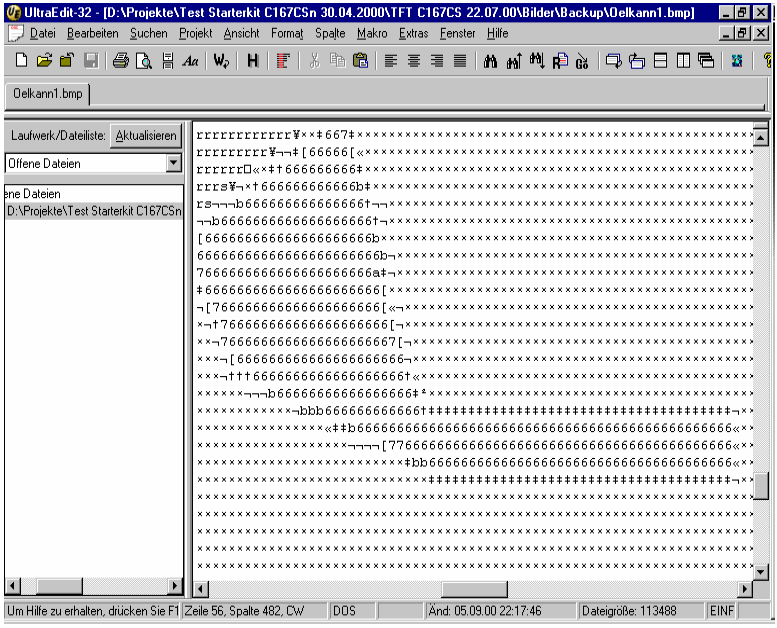

Figure 4 Formated with Ultra Edit

The output file for a 480 * 234 * 3 dot matrix TFT display contains 128 Kbyte of data, which are to be split into two arrays of 64 Kbyte each (max. size of a huge const array is limited to 64 Kbyte).

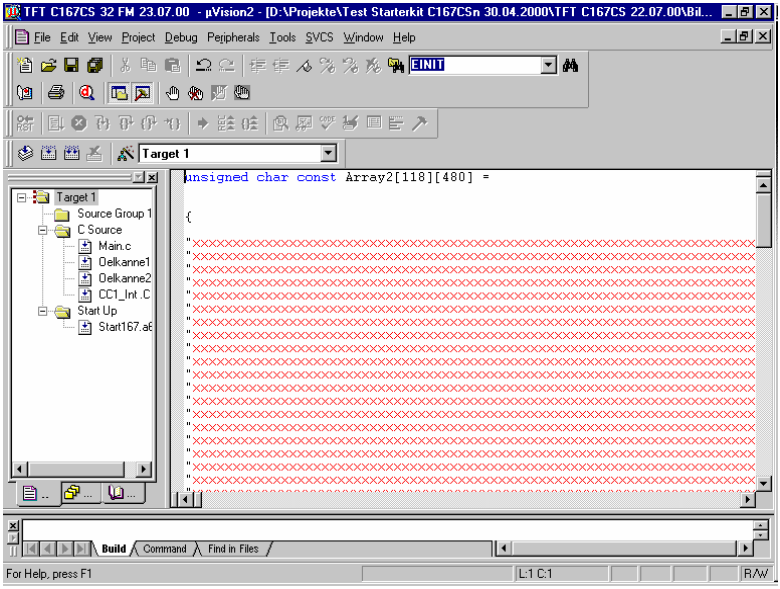

Figure 5 The output file for a 480*234*3 dot matrix TFT display

Here you can see the a part of data regarding the nozzle of the oil can.

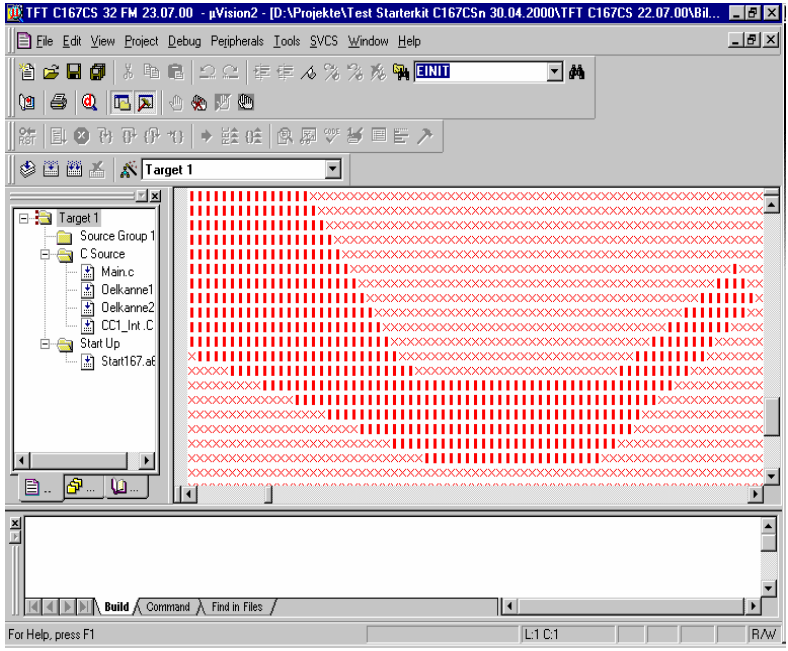

Figure 6 The nozzle of the oil can

1.2 Blockdiagramm of the display configuration

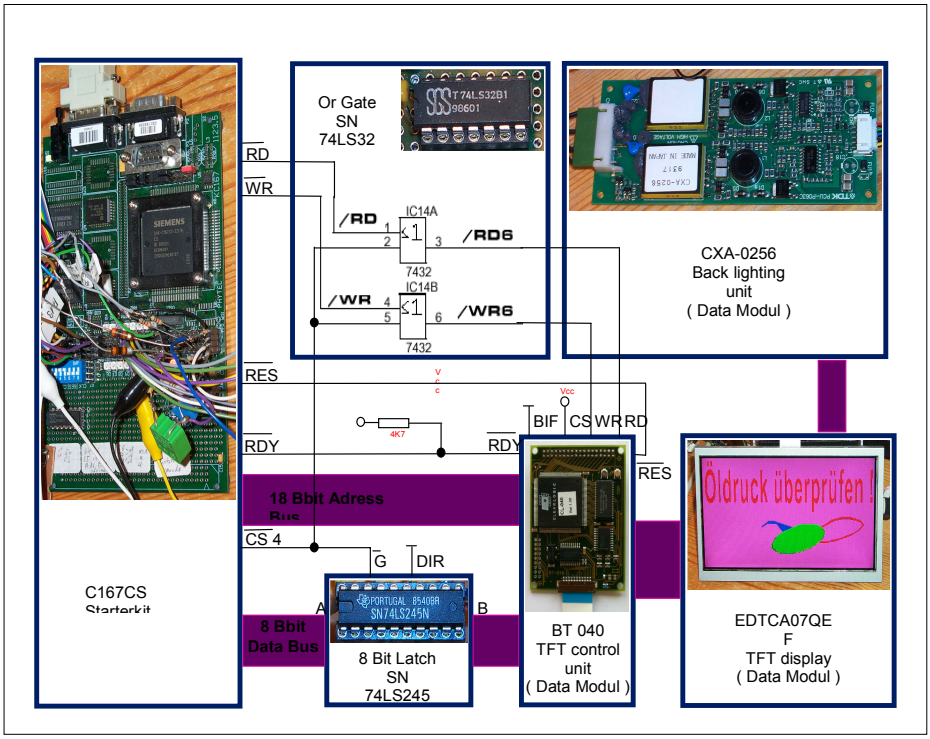

Figure 7 block diagram of the display configuration

2 Hardware description

2.1 KITCON connector C167CS Starterkit

KitCON interface connection – Connector to the BT 040 unit

Table 1 Pindescription C167CS-32FM to BT 040

PIN	Des	BT 040	PIN	Des	BT 040	PIN	Des	BT 040	PIN	Des	BT 040
1	Vcc	34	2	Vcc	34	3	GND	35	4	GND	35
5	D0	1	6	D2	3	7	D4	5	8	D6	7
9	D1	2	10	D3	4	11	D5	6	12	D7	8
13			14			15			16		
17			18			19			20		
21	A0	9	22	A2	11	23	A4	13	24	A6	15
25	A1	10	26	A3	12	27	A5	14	28	A7	16
29	A8	17	30	A10	19	31	A12	21	32	A14	23
33	A9	18	34	A11	20	35	A13	22	36	A15	24
37	A16	25	38			39			40		
41			42			43			44		
45	RD	*1	46			47			48	RES#	27/NRES
49	WR	*1	50			51			52		
53			54			55	CS4#	*1	56		
57			58			59			60		
61			62			63			64		
65			66			67			68		
69			70			71			72		

73			74			75			76		
77			78			79			80		
81			82			83			84		
85			86			87			88		
89			90			91			92		
93			94			95			96		
97			98			99			100		
101			102			103			104		
105			106			107			108		
109			110			111			112	**RDY#**	32/NRDY
113			114			115			116		
117			118			119			120		
121			122			123			124		
125			126			127			128		
129			130			131			132		
133			134			135			136		
137			138			139			140		
141			142			143			144		
145			146			147			148		
149			150			151			152		

Connection of a couple of neccessary signals

BIF (BT 040) go to Intel Mode -> Low -> GND via Pull Down resistor 10 KΩ

RES (BT 040) aktive connected -> Low -> GND vis Pull Down resistor 10 KΩ

CS (BT 040) aktive connected -> High -> via Portpin P 2.7 (SW enable TFT, no CS in origin sense)

LCDCLK (BT 040) already in BT 040 generated, leave unconnected

RDY (BT 040) aktive connected -> Low -> GND via Pull Down resistor 10 KΩ

NRDY (BT 040) via Pull Up resistor 4,7 KΩ against Vcc

Connection CS 4# with RD# and WR# (C167CS-32FM) to RD# and WR# (BT 040)

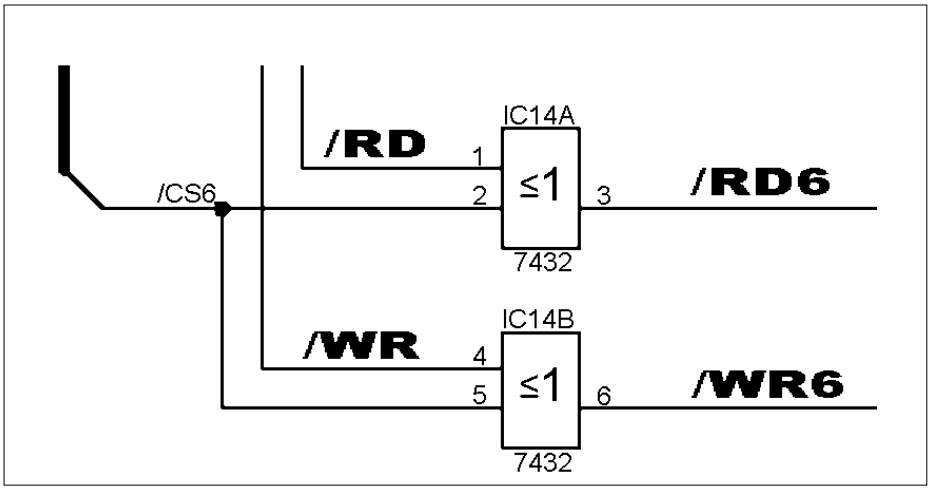

Figure 8 Connection CS 4# with RD# and WR#

Connecting BT 040 to the PCB adapting unit

Flat cabel has to be adjusted with the right side regarding the dedicated connector, otherwise short circuits over will be the consequence!

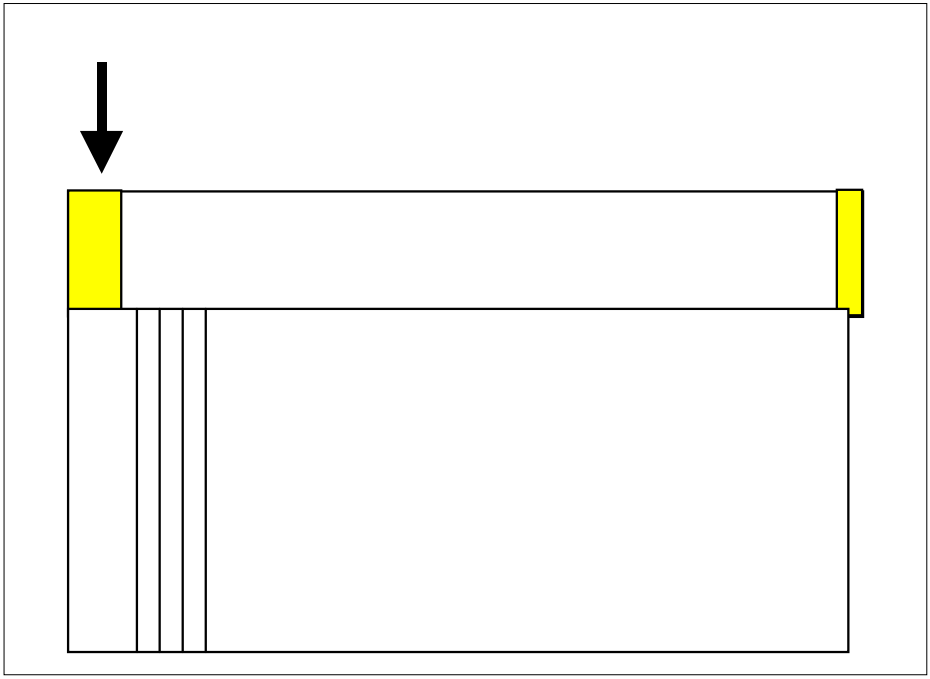

Figure 9 Connecfing BT 040

Note: Please connect the cable conclusive!

2.2 Connection of the TFT controller card BT 040

Table 2 Connection of the controller card

Pin	Function	Description
1	D0	Data bit 0, bi-directional
2	D1	Data bit 1, bi-directional
3	D2	Data bit 2, bi-directional
4	D3	Data bit 3, bi-directional
5	D4	Data bit 4, bi-directional
6	D5	Data bit 5, bi-directional
7	D6	Data bit 6, bi-directional
8	D7	Data bit 7, bi-directional
9	A0	Address bit 0, Input
10	A1	Address bit 1, Input
11	A2	Address bit 2, Input
12	A3	Address bit 3, Input
13	A4	Address bit 4, Input
14	A5	Address bit 5, Input
15	A6	Address bit 6, Input
16	A7	Address bit 7, Input
17	A8	Address bit 8, Input
18	A9	Address bit 9, Input
19	A10	Address bit 10, Input
20	A11	Address bit 11, Input
21	A12	Address bit 12, Input
22	A13	Address bit 13, Input
23	A14	Address bit 14, Input
24	A15	Address bit 15, Input
25	A16	Address bit 16, Input
26	RES	Reset, Input, High-active
27	NRES	Reset; Input, Low-active
28	CS	Chip Select, Input, Low-active
29	RD	Read Signal, Input, (see setting Bus-Interface)

30	WR	Write Signal, Input, Low-active, (see setting Bus-Interface)
31	RDY	Ready Signal, Output, High active
32	NRDY	Ready Signal, Output, Low active, Open Collector
33	BIF	Bus-Interface, Input, (see setting Bus-Interface)
34	VCC	Power Supply, +5V
35	GND	Ground
36	LCDCLK	LCD-Clock, Input / Output Attention: Use only when the oscillator is not equipped! (see LCD-Clock)

2.3 Description of the Latch between the BT 040 / (SN74LS245N)

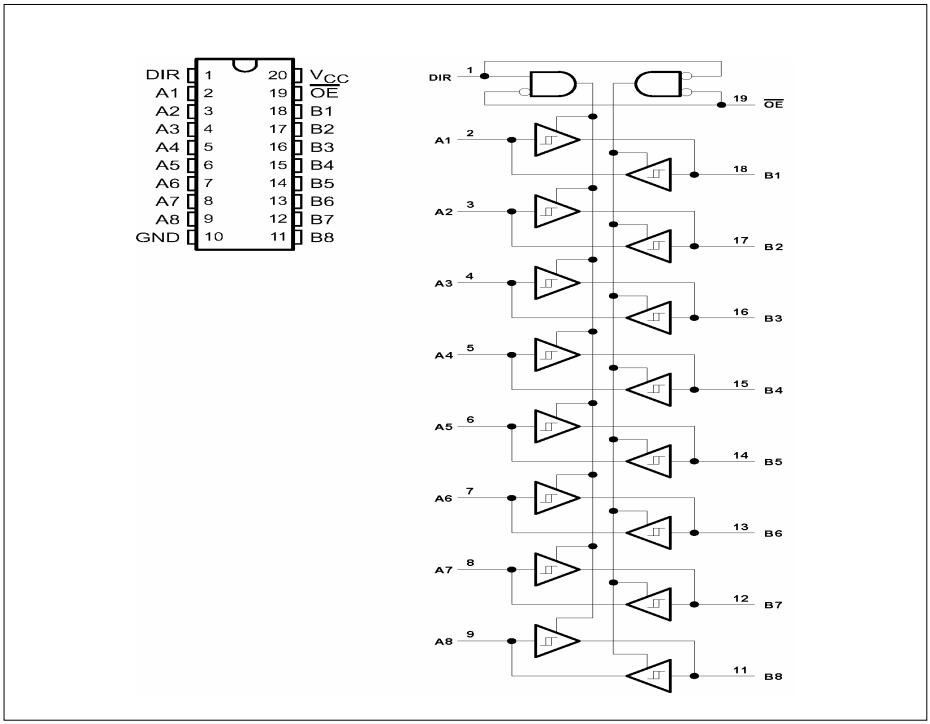

Figure 10 The Latch between the BT 040 / SN74LS245N

2.4 Hardware description EDTCA07QEF / General specification data

Table 3 General specification data EDTCA07QEF

Characteristic Item	Specification
1. Display technology	a-Si active-matrix
2. Display mode	NW (normally white)
3. Module outer dimension [1)]	170.0 (W) * 106.8 (H) * 8.3(D) mm
4. Effective display area	155.52 * 87.28 mm
5. Number of dots	480 (W) * 3 (RG3) * 234 (H)
6. Color-filter-array	RGB vertical stripe
7. Weight	205 ± 10g
8. Backlight	CCFL with 3 wave-length spectrum straight type 2
9. Front surface treatment	AG coat (Haze ratio 12%) (with WV film)
10. Polarizer protective sheet	None
11. Appearance	There are no remarkable defects
12. Metal frame condition	Not be connected to inner circuit

1) Detailed dimensions are shown per attached drawing

2.5 Hardware description EDTCA07QEF / Pixel arrangement regarding the data representation

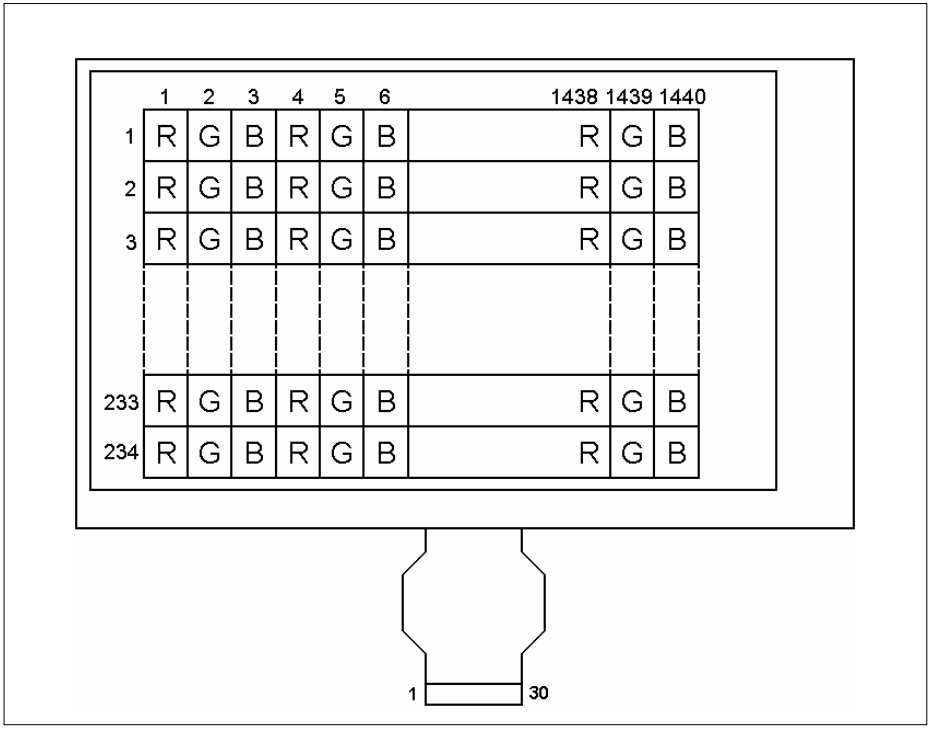

Figure 11 Pixel arrangement and I/O interface pin assignment

2.6 Hardwaredescription EDTCA07QEF / View description of the display hardware

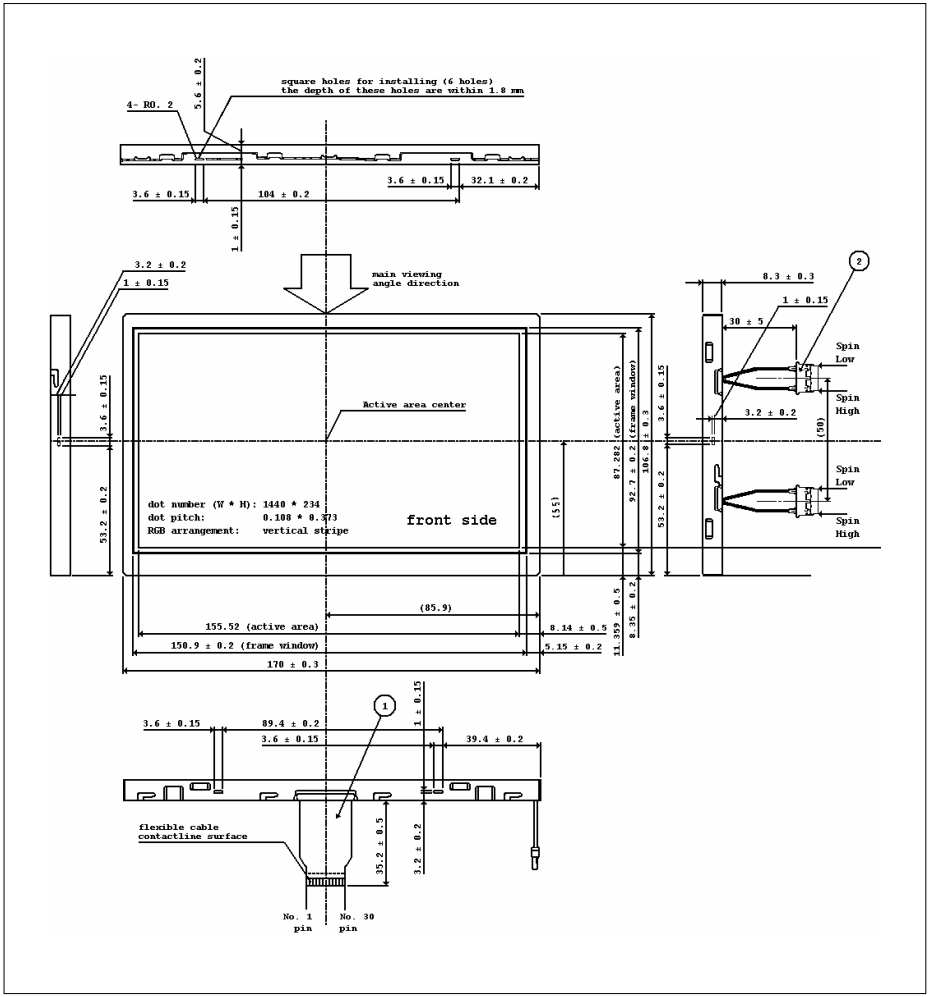

Figure 12 Description of the display hardware

2.7 Hardwaredescription Adresse space and Address range

The TFT Display start address should be 10:0000 h, the address range 128 Kbyte. The BUSCON 4 Register will be used, because it is freely usable on the Starterkit. The BT 040 should be accessed more or less like a static RAM with asyncronous ready.

The resulting window size is given with RGSZ = 5 (Manual C167CS V2.0 Page 8-21).

```
ADDRSEL4_RGSZ   SET     5
```

The remaining number of address lines are A23 to A17.

Table 4 Conversion Hex -> Decimal

1h	0h	0h	0h	0h	0h
0001	0000	0000	0000	0000	0000
A23	A19	A15	A11	A7	A3

Bit field RGSZ	Resulting Window Size	Relevant Bits (R) of Start Addr. (A12 …)
0 0 0 0	4 KBytes	R R R R R R R R R R R R
0 0 0 1	8 KBytes	R R R R R R R R R R R x
0 1 0 1	128 KBytes	R R R R R R R x x x x x
		A23 A21 A19 A17 A15 A13
1 0 0 1	2 MBytes	R R R x x x x x x x x x
1 0 1 0	4 MBytes	R R x x x x x x x x x x
1 0 1 1	8 MBytes	R x x x x x x x x x x x
1 1 x x	Reserved	

So we have to enter the below mentioned value into the RGSAG area.

ADDRSEL4 **SFR ($FE1E_H$ / CF_H)** **[Reset Value: 0000_H]**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				RGSAD								RGSZ			
				rw								rw			
0	0	0	1	0	0	0	0	0	0	0	0				

The entire value inserted into ADDRSEL is 1005h.

The values which are to be inserted into BUSCON 4 you can see below (2 Waitstates asyncronous ready, bus typ -> 8 bit non mux).

```
; Bus Configuration Register 4
BUSCON4_ALECTL4         SET     0
BUSCON4_BTYP            SET     0
BUSCON4_BUSACT4         SET     1
BUSCON4_CSREN4          SET     0
BUSCON4_CSWEN4          SET     0
BUSCON4_MCTC            SET     13
BUSCON4_MTTC4           SET     0
BUSCON4_RDYEN4          SET     1
BUSCON4_RWDC4           SET     0
```

(S- 8-18 Manual V2.0 C167CR)

BUSCON4
Bus Control Register 4 **SFR ($FF1A_H$ / $8D_H$)** **[Reset Value: 0000_H]**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSW EN4	CSR EN4	-	RDY EN4	-	BUS ACT 4	ALE CTL 4	-	BTYP		MTT C4	RWD C4	MCTC			
rw	rw	-	rw	-	rw	rw	-	rw		rw	rw	rw			

3 Software

3.1 Specialities regarding the Start UP file

The PLL factor has to be arranged by implementing a sequence concerning the RSTCON register.

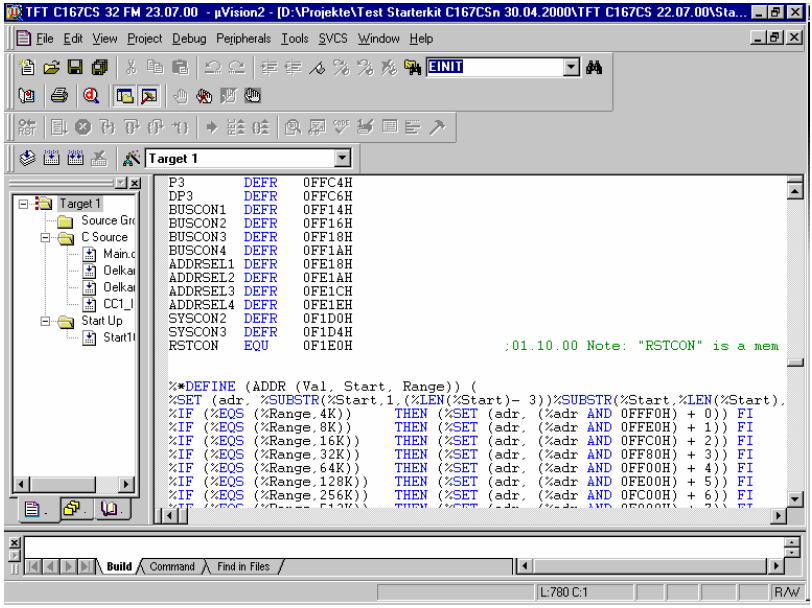

Figure 13

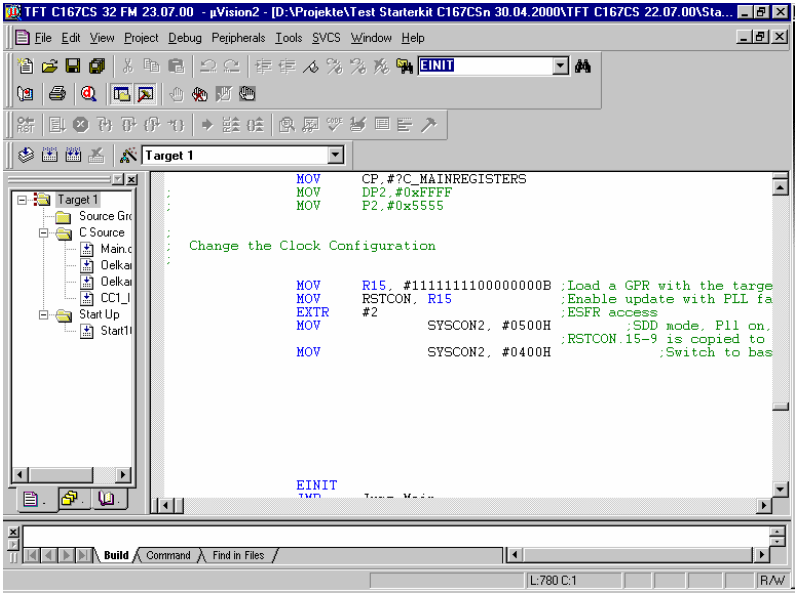
TFT C167CS 32 FM 23.07.00 - µVision2 - [D:\Projekte\Test Starterkit C167CSn 30.04.2000\TFT C167CS 22.07.00\Sta...
File Edit View Project Debug Peripherals Tools SVCS Window Help
EINIT
Target 1
Target 1
Source Gr
C Source
Main.c
Oelkar
Oelkar
CC1_I
Start Up
Start1I
MOV CP,#?C_MAINREGISTERS
MOV DP2,#0xFFFF
MOV P2,#0x5555
; Change the Clock Configuration
MOV R15, #1111111100000000B ;Load a GPR with the targe
MOV RSTCON, R15 ;Enable update with PLL fa
EXTR #2 ;ESFR access
MOV SYSCON2, #0500H ;SDD mode, Pll on,
;RSTCON.15-9 is copied to
MOV SYSCON2, #0400H ;Switch to bas
EINIT
Build Command Find in Files
L:780 C:1
R/W

Figure 14

3.2 First loop for the two 64 K data blocks in main.c

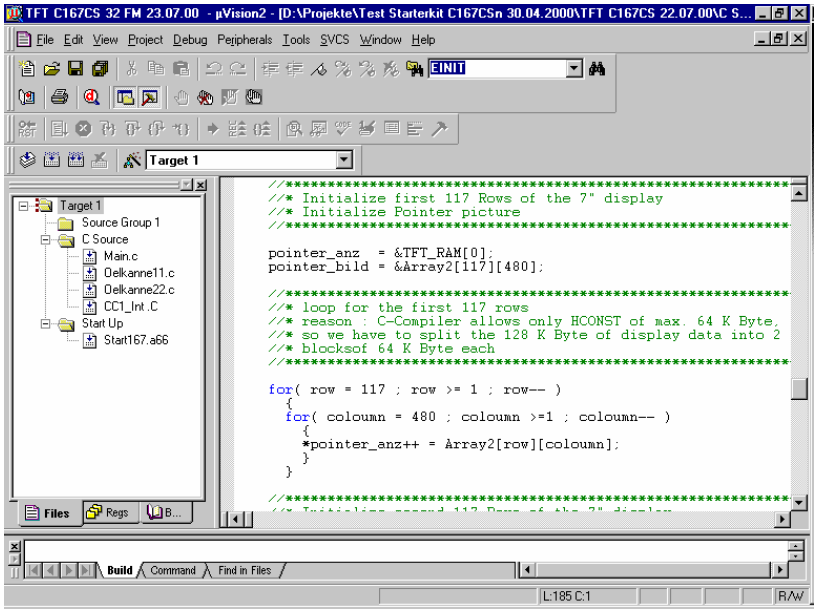

Figure 15

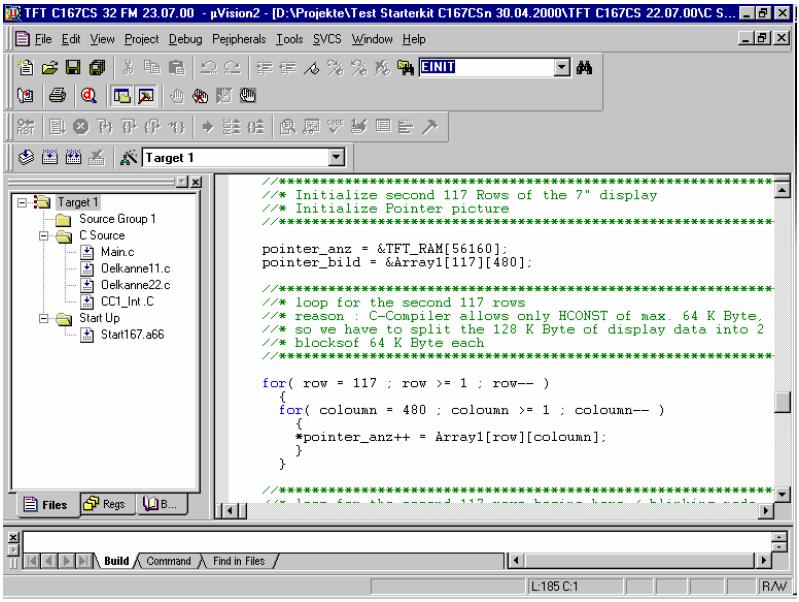
TFT C167CS 32 FM 23.07.00 - µVision2 - [D:\Projekte\Test Starterkit C167CSn 30.04.2000\TFT C167CS 22.07.00\C S...
File Edit View Project Debug Peripherals Tools SVCS Window Help
EINIT
Target 1
Target 1
Source Group 1
C Source
Main.c
Oelkanne11.c
Oelkanne22.c
CC1_Int .C
Start Up
Start167.a66
Files
Regs
B...
//* Initialize second 117 Rows of the 7" display
//* Initialize Pointer picture
pointer_anz = &TFT_RAM[56160];
pointer_bild = &Array1[117][480];
//* loop for the second 117 rows
//* reason : C-Compiler allows only HCONST of max. 64 K Byte,
//* so we have to split the 128 K Byte of display data into 2
//* blocksof 64 K Byte each
for(row = 117 ; row >= 1 ; row--)
{
for(coloumn = 480 ; coloumn >= 1 ; coloumn--)
{
*pointer_anz++ = Array1[row][coloumn];
}
}
Build
Command
Find in Files
L:185 C:1
R/W

Figure 16

3.3 Second loop for the two 64 K data blocks realizing the blinking mode in main.c

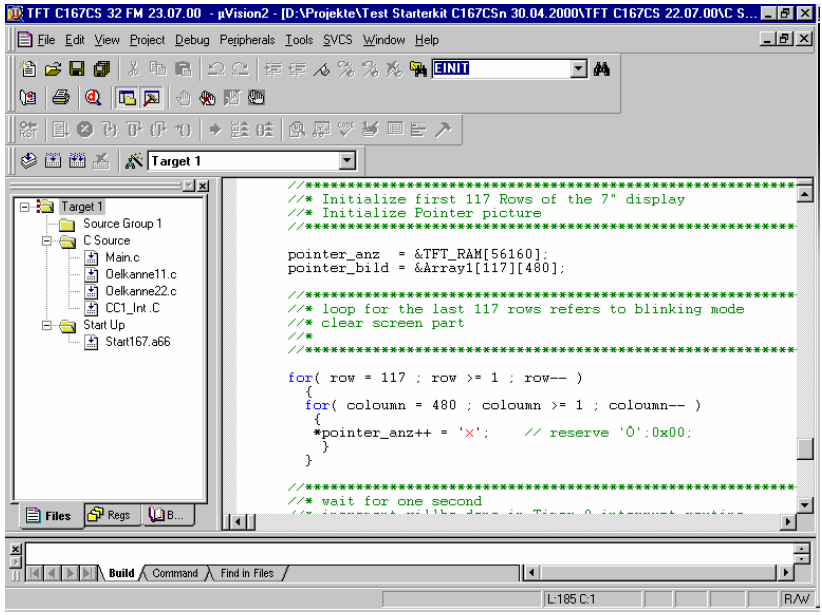

Figure 17

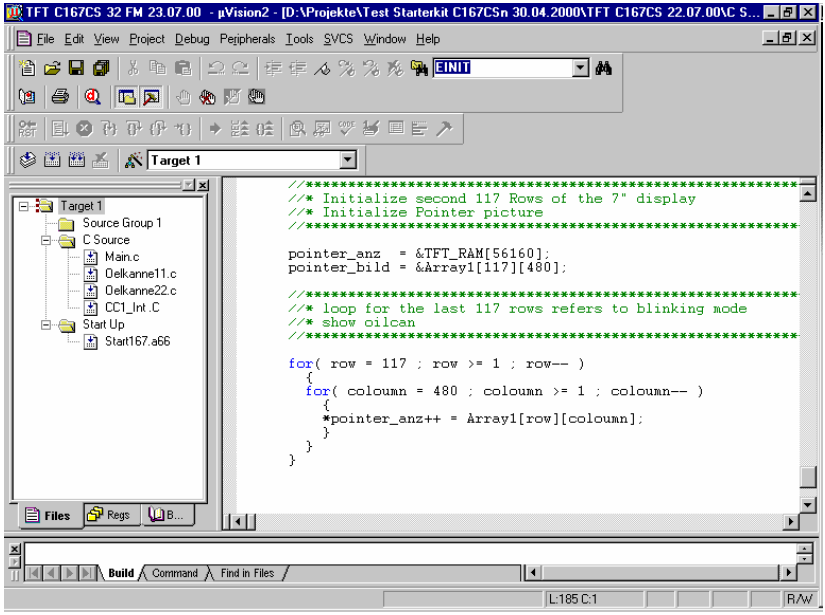
TFT C167CS 32 FM 23.07.00 - µVision2 - [D:\Projekte\Test Starterkit C167CSn 30.04.2000\TFT C167CS 22.07.00\C S...
File Edit View Project Debug Peripherals Tools SVCS Window Help
EINIT
Target 1
Target 1
Source Group 1
C Source
Main.c
Oelkanne11.c
Oelkanne22.c
CC1_Int.C
Start Up
Start167.a66
Files Regs B...
//* Initialize second 117 Rows of the 7" display
//* Initialize Pointer picture
pointer_anz = &TFT_RAM[56160];
pointer_bild = &Array1[117][480];
//* loop for the last 117 rows refers to blinking mode
//* show oilcan
for(row = 117 ; row >= 1 ; row--)
{
for(coloumn = 480 ; coloumn >= 1 ; coloumn--)
{
*pointer_anz++ = Array1[row][coloumn];
}
}
}
Build Command Find in Files
L:185 C:1
R/W

Figure 18

3.4 Generating a one second blinking time in main.c

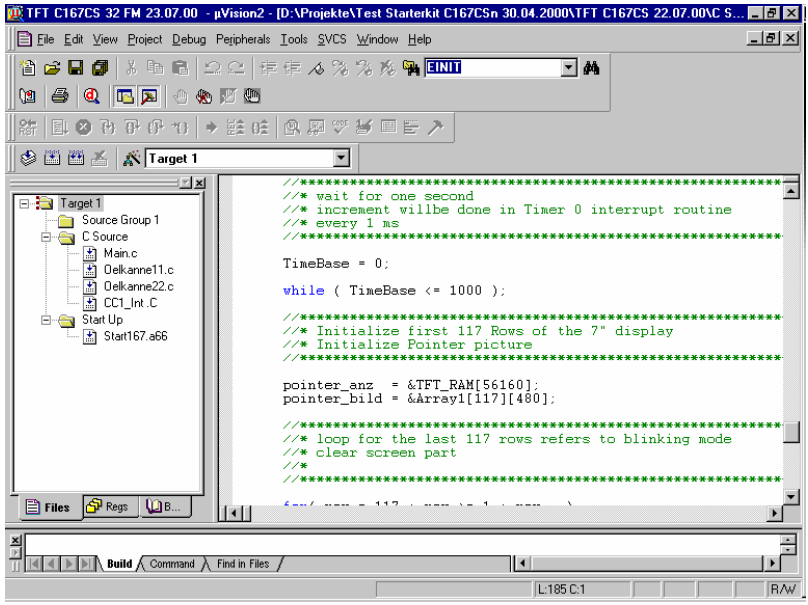

Figure 19

3.5 Timer 0 initialization routine

```
//***************************************************************************
// @Module        CC1
// @Filename      CC1_Int .C
//
//---------------------------------------------------------------------------
// @Controller    Infineon C167CS-32FM
//
// @Compiler      KEIL C166
//***************************************************************************
// @Capture Compare Channel / Timer 0 initialization
//===========================================================
// Timer 0      :    Initialisierung 10 ms Interrupt
// Goal         :    Key active checking
// Date cahange :    21.08.00
// Name         :    Kattw.
//===========================================================
//***************************************************************************


//***************************************************************************
// @Project Includes
//***************************************************************************

#include "MAIN.H"

//***************************************************************************
//* definition of the portpin for the testpin 1 KHz P3.8
//***************************************************************************

sbit  P3_8  = P3^8;
sbit  DP3_8 = DP3^8;

// USER CODE END


//***************************************************************************
// @Global Variables
//***************************************************************************



//***************************************************************************
// @Function      void CC1_Initialization(void)
```

```
//
//---------------------------------------------------------------------
// @Description        This function initializes the CAPCOM 1
//                     component.It effects all necessary configurations
//                     of the SFR, depending on the selected operating
//                     mode. The configuration determines whether the
//                     CAPCOM interrupts are to be released, and the
//                     priority of the released interrupt.
//
//---------------------------------------------------------------------
// @Returnvalue   none
//
//---------------------------------------------------------------------
// @Parameters    none
//
//---------------------------------------------------------------------
// @Date          02.10.00 20:40:18
//
//*********************************************************************

void CC1_Timer0_Initialization(void)
{
  T01CON = 0x0007;

  //*******************************************************************
  //* reloadvalue defined that 1 KHz that Timer 0 interrupt willbe
  //* called every 1 ms
  //* ---- Timer 0 Configuration -------------
  //* timer 0 works in timer mode
  //* prescaler factor is 1024
  //*******************************************************************

  T0     = 0xFFF6;    // load timer 0 register
  T1     = 0x0000;    // load timer 1 register
  T0REL  = 0xFFF6;    // load timer 0 reload register
  T1REL  = 0x0000;    // load timer 1 reload register

  //*******************************************************************
  //*  enable timer 0 interrupt
  //*  timer 0 interrupt priority level(ILVL) = 6
  //*  timer 0 interrupt group level (GLVL) = 3
  //*******************************************************************

  T0IC = 0x005B;

  //*******************************************************************
  //* P2.0 is used as output for CAPCOM channel 0
```

```
  //* P2.8 is used as output for CAPCOM channel 8
  //* initializes port 2 for the capture/compare channels
  //*********************************************************************

  P2    |= 0x0000;      //  set port data register
  ODP2  |= 0x0000;      //  set port open drain control register
  DP2   |= 0x0101;      //  set port direction register


  //*********************************************************************
  //* ---- Capture Compare Channel 0 -------------
  //*  compare mode 3:  set output pin CC0IO on each match
  //*  CC0 allocated to timer T0
  //*********************************************************************

  CCM0 = 0x0007;
  CC0  = 0x0000;  // load CC0 register

  //*********************************************************************
  //* ---- Capture Compare Channel 8 -------------
  //*  compare mode 1:  toggle output pin CC8IO on each match
  //*  CC8 allocated to timer T0
  //*********************************************************************

  CCM2 = 0x0005;

  //*********************************************************************
  //* Comparevalue of CC8 will realize 50 % duty cycle
  //*********************************************************************

  CC8  = 0xFFFA;  // load CC8 register


  //*********************************************************************
  //* timer 0 is running
  //*********************************************************************

  T01CON |= 0x0040;

  //*********************************************************************
  //* define  P3.8 as an output an set portpin to high
  //*********************************************************************

  P3_8    = 1;      //  set port data register for Test Pin
  DP3_8   = 1;      //  set port direction register for Test Pin
}
```

3.6 Timer 0 interrupt routine generating a one second blinking repeat frequency

```
//***************************************************************************
// @Function      void Timer 0 Interrupt(void) interrupt T0INT
//
//---------------------------------------------------------------------------
//***************************************************************************
// @Timer 0 Interrupt routine
//===========================================================
// Timer 0       :    Initialisierung 10 ms Interrupt
// Goal          :    Key active checking
// Date cahange  :    21.08.00
// Name          :    Kattw.
//===========================================================
//***************************************************************************

//***************************************************************************
//* includefiles
//***************************************************************************

#include "MAIN.H"
#include "CC1.H"

//***************************************************************************
//* definition of the portpin for the testpin 1 KHz P3.8
//***************************************************************************

sbit  P3_8  = P3^8;

//***************************************************************************
//* externla data variables
//***************************************************************************

extern unsigned int TimeBase;

//***************************************************************************
//* interrupt routine Timer 0
//***************************************************************************

void Timer_0_Interrupt(void) interrupt T0INT
  {
  //*************************************************************************
  //* increment time variable
  //*************************************************************************
```

```
  TimeBase++;            // will be used in main


//*************************************************************************
  //* toggle P3.8 with an frequency of 1 KHz
  //*************************************************************************

  P3_8 = !P3_8;          // test rectangle 1 KHz
}
```

3.7 Main program

```
//***************************************************************************
// @Module        Main
// @Filename      MAIN.C
// @Project       TFT V 1.0
//---------------------------------------------------------------------------
// @Controller    Infineon C167CS-32FM DA Step
//
// @Compiler      KEIL C166 V2.05
//
// @Description   This file contains the Project initialization
function.
//
//---------------------------------------------------------------------------
// @Date          29.09.00
//
//***************************************************************************

//***************************************************************************
//*@Note
//*
//*
//* BUSCON 4 will be used for addressing TFT with asynchronous ready,
//* 3 waitstates, 128 KByte Adressraum
//*
//***************************************************************************
//*/
//***************************************************************************
// @Project Includes
//***************************************************************************

#include "MAIN.H"

//***************************************************************************
//* external background picture arrays
//***************************************************************************

extern unsigned char const Array1[118][480];
extern unsigned char const Array2[118][480];

//***************************************************************************
// @Global Variables
//***************************************************************************
```

```
unsigned char xhuge TFT_RAM [0x20000];
unsigned char xhuge *pointer_anz;
unsigned char *pointer_bild;

unsigned int  TimeBase = 0;

//***************************************************************************
// @Test Variables
//***************************************************************************

sbit  P3_8  = P3^8;
sbit  DP3_8 = DP3^8;

sbit  P2_8  = P2^8;
sbit  DP2_8 = DP2^8;

sbit  P2_0  = P2^0;
sbit  DP2_0 = DP2^0;

sbit  P2_1  = P2^1;
sbit  DP2_1 = DP2^1;

//***************************************************************************
// @Project initialization
//***************************************************************************

void Project_Init(void)
{
  //*************************************************************************
  // @Timer 0 initialization
  //=========================================================================
  // Timer 0      :    Initialisierung 10 ms Interrupt
  // Goal         :    Key active checking
  // Date cahange :    21.08.00
  // Name         :    Kattw.
  //=========================================================================
  //*************************************************************************

  CC1_Timer0_Initialization();

  P2_8    = 0;     //  set port data register
  DP2_8   = 1;     //  set port data register

  //*************************************************************************
  // @global interrupt enable
  //*************************************************************************
```

```
  IEN = 1;
}


//***************************************************************************
// @Function      void main(void)
//
//===========================================================================
// Timer 0       :    Initialisierung 10 ms Interrupt
// Goal          :    adressing TFT display
// Date cahange  :    21.08.00
// Name          :    Kattw.
//===========================================================================
//***************************************************************************

void main(void)
  {

  //*************************************************************************
  //* define variables for the loops
  //*************************************************************************

  int row, coloumn;

  Project_Init();

  //*************************************************************************
  //* define the value for several portpins
  //*************************************************************************

  P2_0    = 1;      //  set port data register
  P2_1    = 1;      //  set port data register

  //*************************************************************************
  //* define the value for portdirection of several portpins
  //*************************************************************************

  DP2_0   = 0;      //  set port data register
  DP2_1   = 1;      //  set port data register

  //*************************************************************************
  //* define the value for several portpins
  //* only siple version of testing without debruise function
  //*************************************************************************

  while (P2_0 == 1)   // Tastenabfrage P2.0
    {
```

```
  P2_1 = 1;            // Taste gedrückt ?
  }

P2_1 = 0;              // Taste wurde gedrückt !


  //*********************************************************************
  //* Initialize first 117 Rows of the 7" display
  //* Initialize Pointer picture
  //*********************************************************************

  pointer_anz  = &TFT_RAM[0];
  pointer_bild = &Array2[117][480];

  //*********************************************************************
  //* loop for the first 117 rows
  //* reason : C-Compiler allows only HCONST of max. 64 K Byte,
  //* so we have to split the 128 K Byte of display data into 2
  //* blocksof 64 K Byte each
  //*********************************************************************

  for(row = 117 ; row >= 1 ; row--)
    {
    for(coloumn = 480 ; coloumn >=1 ; coloumn--)
      {
      *pointer_anz++ = Array2[row][coloumn];
      }
    }

  //*********************************************************************
  //* Initialize second 117 Rows of the 7" display
  //* Initialize Pointer picture
  //*********************************************************************

  pointer_anz = &TFT_RAM[56160];
  pointer_bild = &Array1[117][480];

  //*********************************************************************
  //* loop for the second 117 rows
  //* reason : C-Compiler allows only HCONST of max. 64 K Byte,
  //* so we have to split the 128 K Byte of display data into 2
  //* blocks of 64 K Byte each
  //*********************************************************************

  for(row = 117 ; row >= 1 ; row--)
    {
    for(coloumn = 480 ; coloumn >= 1 ; coloumn--)
```

```
    {
    *pointer_anz++ = Array1[row][coloumn];
    }
  }

//*********************************************************************
//* loop for the second 117 rows begins here / blinking mode
//*********************************************************************

while(1)

  {

  //*********************************************************************
  //* wait for one second
  //* increment will be done in Timer 0 interrupt routine
  //* every 1 ms
  //*********************************************************************

  TimeBase = 0;

  while (TimeBase <= 1000);

  //*********************************************************************
  //* Initialize first 117 Rows of the 7" display
  //* Initialize Pointer picture
  //*********************************************************************

  pointer_anz  = &TFT_RAM[56160];
  pointer_bild = &Array1[117][480];

  //*********************************************************************
  //* loop for the last 117 rows refers to blinking mode
  //* clear screen part
  //*
  //*********************************************************************

  for(row = 117 ; row >= 1 ; row--)
    {
    for(coloumn = 480 ; coloumn >= 1 ; coloumn--)
     {
     *pointer_anz++ = 'x';     // reserve 'Ô';0x00;
      }
    }

  //*********************************************************************
  //* wait for one second
```

```
      //* increment will be done in Timer 0 interrupt routine
      //* every 1 ms
      //*****************************************************************

      TimeBase = 0;

      while (TimeBase <= 1000);

      //*****************************************************************
      //* Initialize second 117 Rows of the 7" display
      //* Initialize Pointer picture
      //*****************************************************************

      pointer_anz  = &TFT_RAM[56160];
      pointer_bild = &Array1[117][480];

      //*****************************************************************
      //* loop for the last 117 rows refers to blinking mode
      //* show oilcan
      //*****************************************************************

      for(row = 117 ; row >= 1 ; row--)
        {
        for(coloumn = 480 ; coloumn >= 1 ; coloumn--)
          {
          *pointer_anz++ = Array1[row][coloumn];
          }
        }
      }
    while(1);
}
```

3.8 Locator Setting

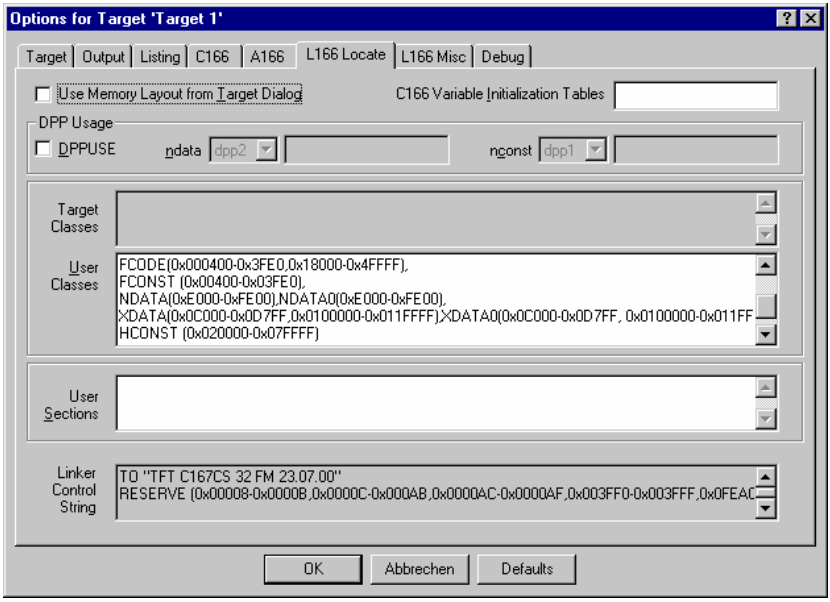

Figure 20 Locator Setting

3.9 BUSCON 4 Settings

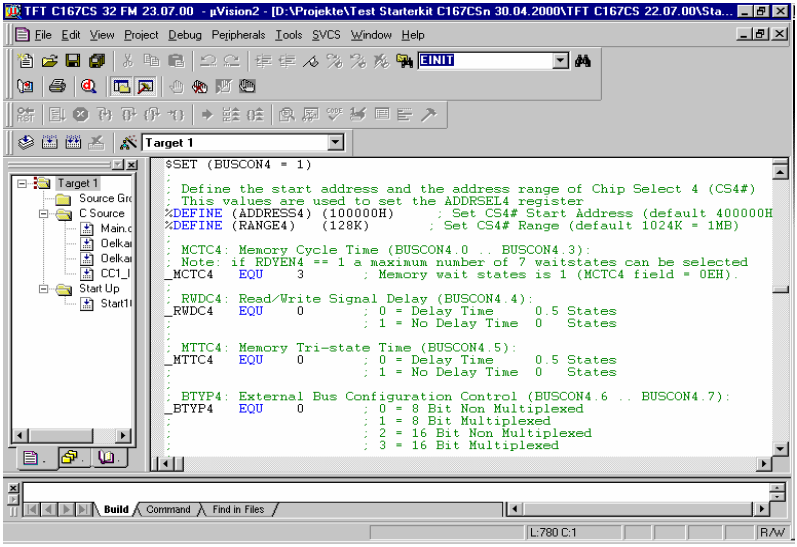

Figure 21

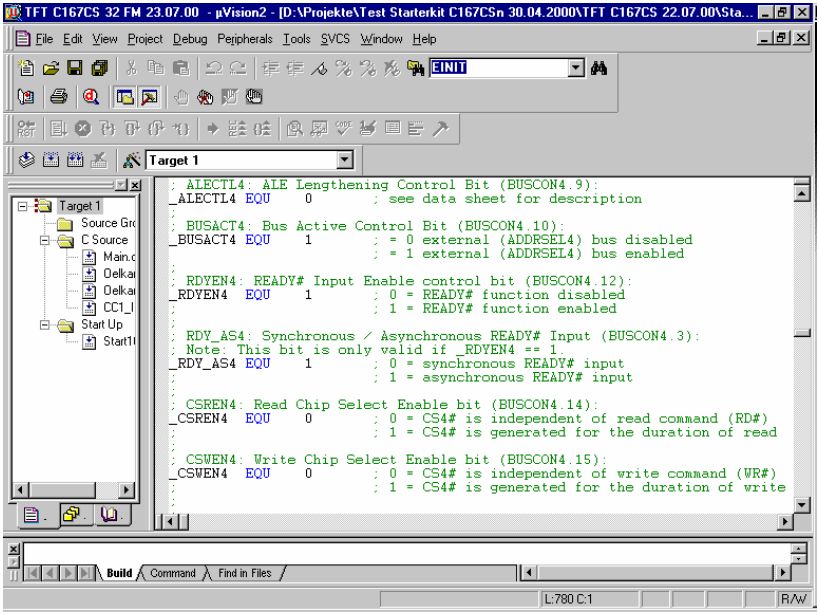
TFT C167CS 32 FM 23.07.00 - µVision2 - [D:\Projekte\Test Starterkit C167CSn 30.04.2000\TFT C167CS 22.07.00\Sta...
File Edit View Project Debug Peripherals Tools SVCS Window Help
EINIT
Target 1
Target 1
Source Gr
C Source
Main.c
Oelkat
Oelkat
CC1_I
Start Up
Start1
; ALECTL4: ALE Lengthening Control Bit (BUSCON4.9):
_ALECTL4 EQU 0 ; see data sheet for description
; BUSACT4: Bus Active Control Bit (BUSCON4.10):
_BUSACT4 EQU 1 ; = 0 external (ADDRSEL4) bus disabled
; = 1 external (ADDRSEL4) bus enabled
; RDYEN4: READY# Input Enable control bit (BUSCON4.12):
_RDYEN4 EQU 1 ; 0 = READY# function disabled
; 1 = READY# function enabled
; RDY_AS4: Synchronous / Asynchronous READY# Input (BUSCON4.3):
; Note: This bit is only valid if _RDYEN4 == 1.
_RDY_AS4 EQU 1 ; 0 = synchronous READY# input
; 1 = asynchronous READY# input
; CSREN4: Read Chip Select Enable bit (BUSCON4.14):
_CSREN4 EQU 0 ; 0 = CS4# is independent of read command (RD#)
; 1 = CS4# is generated for the duration of read
; CSWEN4: Write Chip Select Enable bit (BUSCON4.15):
_CSWEN4 EQU 0 ; 0 = CS4# is independent of write command (WR#)
; 1 = CS4# is generated for the duration of write
Build
Command
Find in Files
L:780 C:1
R/W

Figure 22

3.10 Monitor Driver Settings

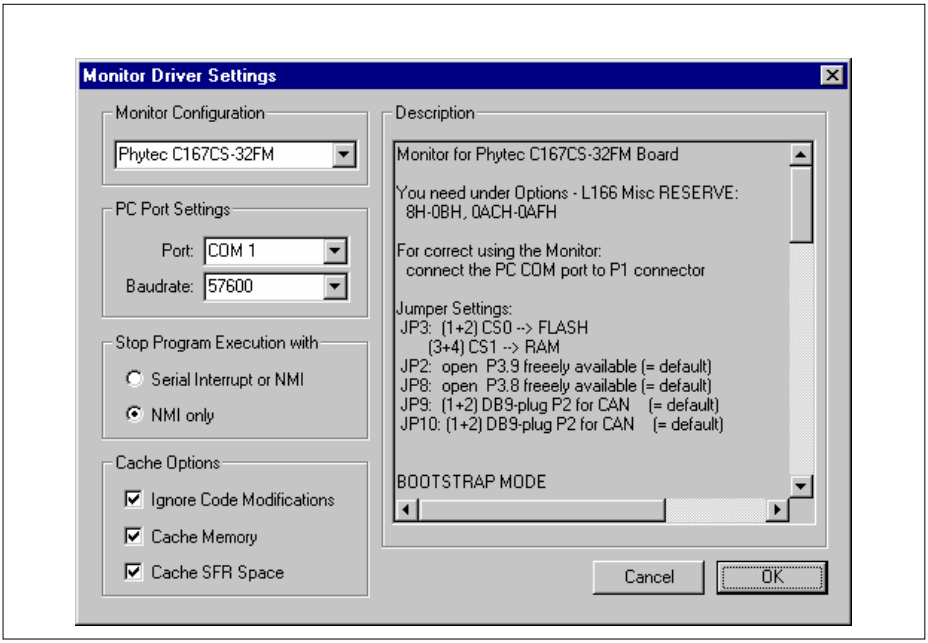

Figure 23 Monitor Driver Settings

4 Miscellaneous

4.1 Pictures of the TFT Driver Module BT 040 (Data Modul)

This Modul delivers the necesary analogue Supply Voltages.

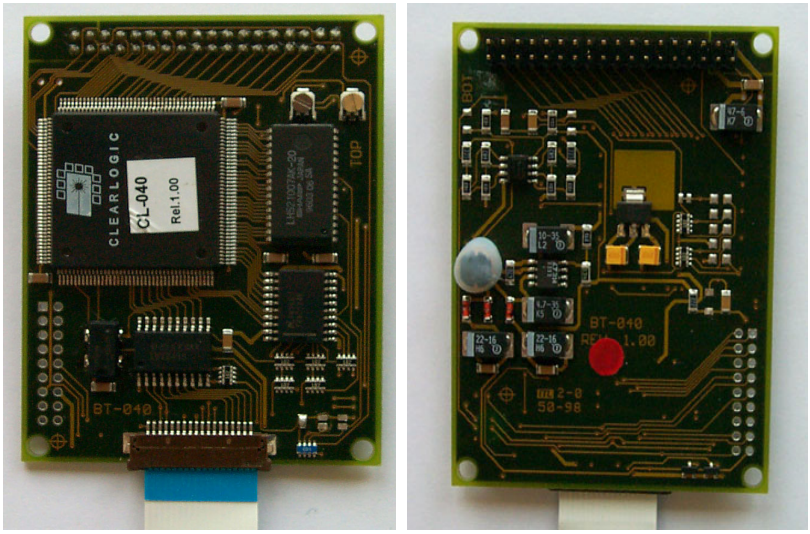

Figure 24 TFT Driver Module BT 040

4.2 Picture of the TFT Display EDTCA07QEF (Data Modul)

The picture with the blinking oilcan as an alarm announcement for the car driver. The oilcan is drawn with Corel Draw, rearranged with Paint Shop Pro and finally confiured into a sufficient data format by using Ultra Edit. This data are adapted ito two arrays of 64 Kbyte each. A SAK C167CS-32FM transfers the data to the TFT control unit of the display (BT 040).

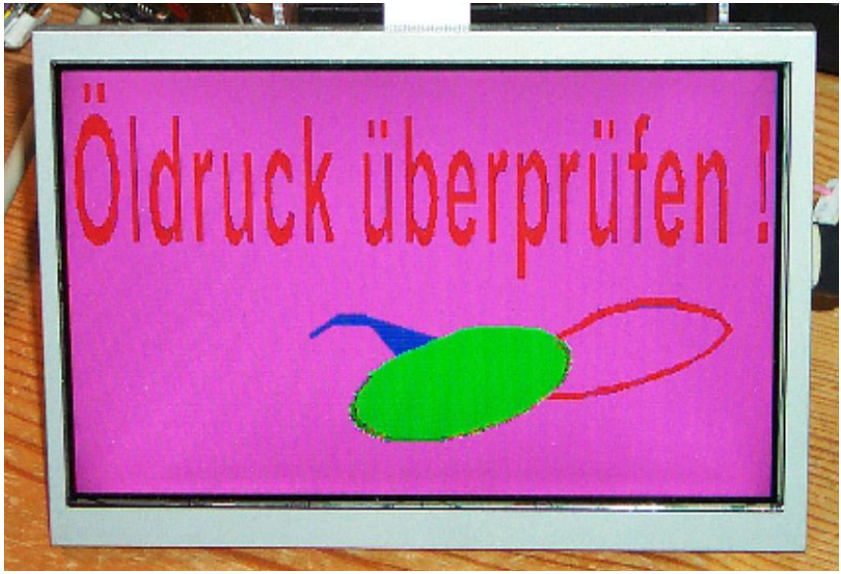

Figure 25 The TFT Display EDTCA07QEF

4.3 Picture of the Backlighting module CXA – 0256 (Data Module)

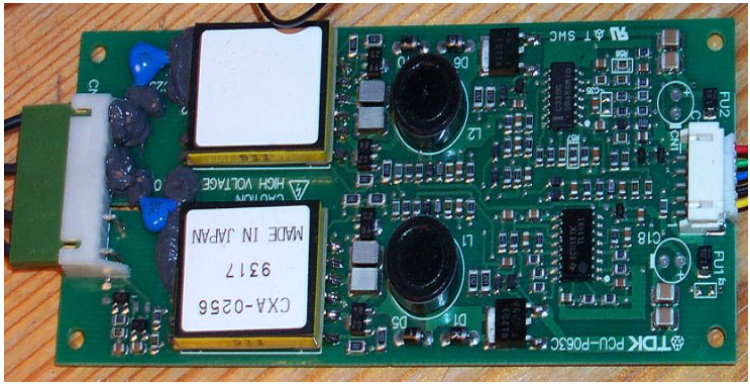

Figure 26 Backlighting module CXA - 0256

4.4 Picture of the Starterkit C167CS-32FM (Phytec)

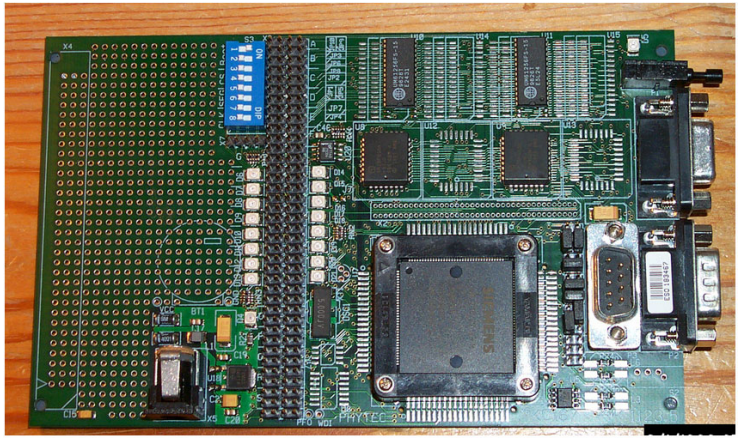

Figure 27 Starterkit C167CS-32FM

4.5 Timing calculations / read timing

Consideration of the BT 040 Read Timings.

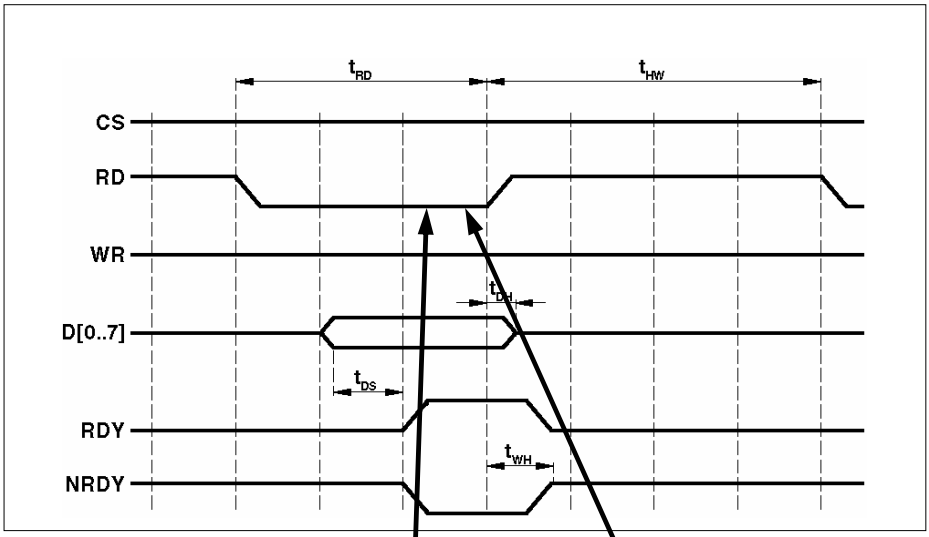

Figure 28 READ-Zyklus, Intel-Interface

Note: RES = Low, NRES = High

Table 5 Timing

Description	Symbol	Minimal	Typical	Maximal	Unit
RD access time	t_{RD}	150	300	1000	ns
Data setup time	t_{DS}	0	-	-	ns
Data hold time	t_{DH}	0	-	30	ns
Ready hold time	t_{WH}	0	-	30	ns
RD repeat time	t_{HW}	100	-	-	ns

Length of Read Signal in reading mode

The length of the low-phase of the RD# signal will be controlled inevitablen via the READY# signal. At the earliest after 160 ns, and not later than 1 us the BT-040 delivers the valid data. From the viewpoint of the C167CS only that time has to be bypassed by waitstates the BT-040 delivers a valid READY# signal. The first assessment of the READY# input will be done (non-extended ALE) with the first rising CLKOUT slope during a bus cycle.

Case 1: previous bus cycle has been no access to the BT-040: RDY or NRDY is stable at status 'not ready'.

⇒ no WS necessay

Case 2: previous bus cycle has been an access to the BT-040 : RDY or NRDY is stable again with status 'not ready' max. 30 ns after the last access.

⇒ 1 memory cycle time waitstate or tristate waitstate or extended ALE necessary @ 20 MHz/non-mux bus.

As well the 'RD repeat time' will be min. 100 ns, so that in case of two each other following accesses @ 20 MHz/Non-Mux bus **a tristate waitstate** or extended ALE is to be used and R/W delay must not be **disabled** (bit RWDCx has to stay = 0).

Note: It is to be considered that the data driver of the BT 040 could be active 30 ns after the rising edge of he RD# signal (will be nor problem if R/W delay is active (t 20/T21)).

Lenght of the write signal in write cycle

The lenght of the low phase of the WR# signal will be controlled inevitable by the READY# signal. At the earliest after 130 ns an at the latest after 600 ns the BT-040 has taken over the write data. Regarding the consideration of the waitstates which will be necessary during a write access in a roughly translated the same arguments are valid as in the read cycle (s.o.).

Which slope is the reference slope?

Reference slope is the CLKOUT; in best case the rising CLKOUT slope corresponds to the falling ALE slope.

t_{37} = 2 TCL + 15, to enshure a secure recognition.

Window length is specified by t_{58}, t_{59}.

Change of the sample point as a function of the number of cycle time waitstates?

The first relevant sample point will be shifted with n Waitstates (n = 0..7) by precise n * 2TCL.

The lenght of the window in relation to the rising edge of CLKOUT remains unchanged specified by t_{58}, t_{59}.

When ready will be recognized?

Generally valid is: will READY# be recognized as ‚high' during the first relevant sample point, wait states will be implemented as long as RAEDY# will be recognized as ‚low'. After that the bus cycle with the next but one falling CLKOUT slope will be stopped.

Is a tristate waitstate to be inserted?

A tristate waitstate is to be inserted on account of read/write repeat time of the BT-040.

Is there any code to be executed out of the internal flash by the SAK C167CS-32FM during waiting on the low signal of the ready pin?

Two cases are to be considered:

- Reading operands: C167CS has to wait until valid data are available. Reason: The µC needs the data for further instruction execution.
- Writing operands: write back of results can be done in parallel to the execution of further instructions out of the internal flash. The C167CS has to wait if he needs further operands from te external bus or from XBUS peripherals.

Is a tristate waitstate necesary?

A tristate waitstate will be necessary regarding the read/write repeat time of the BT-040 (100 ns).

Further the consideration of the ready timings of the C167CS-32FM data sheet SAK C167CS-32FM from 3 / 99.

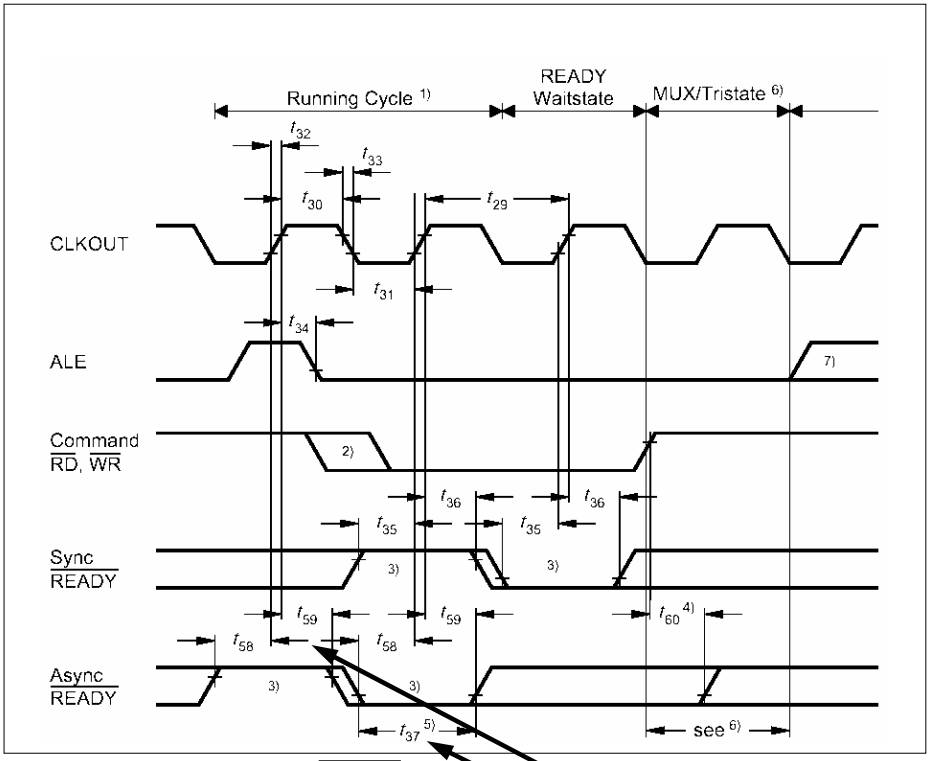

Figure 29 CLKOUT and $\overline{\text{READY}}$

And the appropriate times: Time after rising edge clockout = $t_{37} + t_{59} = 2$ TCL + $t_{58} = 54$ ns.

Table 6 CLKOUT and $\overline{\text{READY}}$

(Operation Conditions apply)

Parameter	Symbol		Max. CPU Clock = 25 MHz		Variable CPU Clock 1 / 2 TCL = 1 to 25 MHz		Unit
			min	max	min	max	
CLKOUT cycle time	t_{29}	CC	40	40	2 TCL	2 TCL	ns
CLKOUT high time	t_{30}	CC	14	-	TCL – 6	-	ns
CLKOUT low time	t_{31}	CC	10	-	TCL – 10	-	ns
CLKOUT rise time	t_{32}	CC	-	4	-	4	ns
CLKOUT fall time	t_{33}	CC	-	4	-	4	ns
CLKOUT rising edge to ALE falling edge	t_{34}	CC	$0 + I_A$	$10 + I_A$	$0 + I_A$	$10 + I_A$	ns
Synchronous $\overline{\text{READY}}$ setup time to CLKOUT	t_{35}	SR	14	-	14	-	ns
Synchronous $\overline{\text{READY}}$ hold time after CLKOUT	t_{36}	SR	4	-	4	-	ns
Asynchronous $\overline{\text{READY}}$ low time	t_{37}	SR	54	-	2 TCL + t_{58}	-	ns
Asynchronous [1] $\overline{\text{READY}}$ setup time	t_{58}	SR	14	-	14	-	ns
Asynchronous [1] $\overline{\text{READY}}$ hold time	t_{59}	SR	4	-	4	-	ns
Asynchronous [2] $\overline{\text{READY}}$ hold time after $\overline{\text{RD}}$, $\overline{\text{WR}}$ high (Demultiplexed Bus)	t_{60}	SR	0	$0 + 2t_A + t_C + t_F$ [2]	0	TCL – 20 + $2t_A + t_C + t_F$ [2]	ns

1) These timings are given for test purposes only, in order to assure recognition at a specific clock edge.

2) Demultiplexed bus is the worst case. For multiplexed bus 2TCL are to be added to the maximum values. This adds even more time for deactivating $\overline{\text{READY}}$.
The $2t_A$ and t_C refer to the next following bus cycle, t_F refers to the current bus cycle.
The maximum limit for t_{60} must fullfilled if the next following bus cycle is $\overline{\text{READY}}$ controlled.

http://www.infineon.com

Published by Infineon Technologies AG