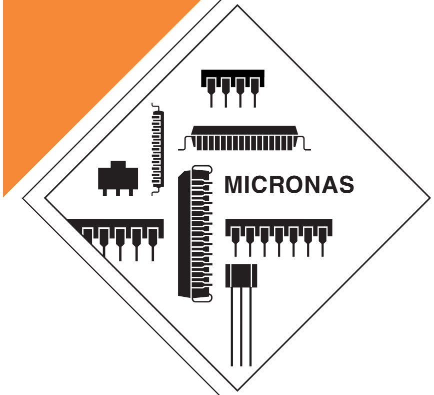


PRELIMINARY DATA SHEET

CCZ 3005K
Central Control Unit

Edition June 28, 2000
6251-471-1PD

MICRONAS

Contents

Page	Section	Title
4	**1.**	**Introduction**
4	1.1.	Features
5	**2.**	**Functional Description**
5	2.1.	CPU
5	2.2.	ROM
5	2.3.	RAM
5	2.4.	Clock Generator
6	2.5.	Control Register
6	2.6.	Reset Function
8	2.7.	Watchdog
9	2.8.	Ports P0–0 to P3–7
10	2.9.	6-bit DACs PWM0 to PWM5
10	2.10.	14-bit DAC PWM6
11	2.11.	I^{2}C and IM-Bus Interface
16	2.12.	A/D Converter
17	2.13.	Closed Caption Acquisition
17	2.13.1.	Video Input
18	2.13.2.	Closed Caption Data Detection
21	2.13.3.	Gate and Window Logic
25	2.14.	OSD
25	2.14.1.	Summary of OSD Features
25	2.14.2.	Fonts
25	2.14.3.	OSD Window
26	2.14.4.	Colors
26	2.14.4.1.	OSD Attribute ‘COLOR’
26	2.14.4.1.1.	Attribute ‘Transparent’
26	2.14.4.2.	Available Colors
27	2.14.4.3.	Color Palette Programming
27	2.14.4.4.	Color Palette Hardware
28	2.14.5.	Fast Blank Output
28	2.14.6.	Half-Video Output
29	2.14.7.	Using OSD
31	2.14.8.	OSD Attributes
32	2.14.9.	Font Definition
32	2.14.10.	Soft-Scroll
33	2.15.	Cursor
33	2.15.1.	Cursor Definition
34	2.15.2.	Cursor Position
34	2.15.3.	Moving and Changing the Cursor
34	2.15.4.	Cursor Control Bits
35	2.15.5.	Cursor DMA
36	2.16.	H&V Sync Generator
38	2.17.	Infrared Input
40	2.17.1.	Infrared Detection Status
40	2.17.2.	Infrared Detection Control
40	2.17.3.	Sample Times

Contents, continued

Page	Section	Title
41	2.18.	Timer
41	2.19.	Interrupt System
42	**3.**	**Specifications**
42	3.1.	Outline Dimensions
42	3.2.	Pin Connections and Short Descriptions
45	3.3.	Pin Descriptions
47	3.4.	Pin Configuration
48	3.5.	Pin Circuits
50	3.6.	Electrical Characteristics
50	3.6.1.	Absolute Maximum Ratings
51	3.6.2.	Recommended Operating Conditions
52	3.6.3.	Recommended Crystal Characteristics
53	3.6.4.	DC Characteristics
54	3.6.5.	DC Parameters I^2C-Bus Master Interface
54	3.6.6.	A/D Converter Characteristics
55	**4.**	**Definitions**
55	4.1.	I/O Definitions
56	**5.**	**Register Description**
74	**6.**	**Appendix A: Closed Caption**
74	6.1.	The Closed Caption Standard
74	6.1.1.	Data Transmission Format
76	6.2.	Closed Caption Decoder
76	6.2.1.	Operating Modes
76	6.2.2.	Screen Format
76	6.2.2.1.	Text Mode
77	6.2.2.2.	Caption Mode
79	6.2.3.	Presentation Format
79	6.2.4.	Character Format
80	6.2.5.	Character Attributes
80	6.2.6.	Control Codes
85	6.2.7.	Data Rejection
85	6.2.8.	Automatic Display Enable/Disable
85	6.2.8.1.	Enable Logic and Timing
85	6.2.8.2.	Disable Logic and Timing
86	**7.**	**Appendix B: Pin Configuration of PGA Package**
87	7.1.	Pin Connections in CPGA132F Package
88	**8.**	**Data Sheet History**

Central Control Unit
Single Chip Microcontroller with Embedded Closed Caption Decoder

1. Introduction

The Central Control Unit CCZ 3005K is an integrated circuit designed in CMOS technology and housed in a 52-pin Plastic Shrink Dual-In-Line Package. It is used as a single-chip TV controller with embedded Closed Caption Decoder and On-Screen Display.

1.1. Features

- 6 MHz 65C02 CPU, 12 MHz crystal
- 52-pin PSDIP package
- on-chip oscillator
- clock generator with programmable frequency
- 62 KBytes internal ROM
- 1536 Bytes internal RAM
- Closed Caption Decoder
- programmable TV-line detector
- 2 selectable H_{SYNC} inputs
- 2 programmable H & V sync outputs
- free running H & V sync generator for stable OSD
- full-screen OSD with separate 24*24 pixel Cursor controls RGB and Fast Blank outputs
- color palette: 8 out of 64 different colors programmable
- soft-scroll, underline, flash, italics
- Half-Video control output
- I^2C/IM-bus master interface
- six 6-bit D/A converters (PWM)
- single 14-bit D/A converter (PWM) for voltage synthesizer
- up to 29 port lines
- 8-bit A/D converter (6-bit precision) with 5 multiplexed inputs
- infrared input hardware supporting software decoding
- free-running timer generating interrupts
- power-on and clock supervision
- watchdog

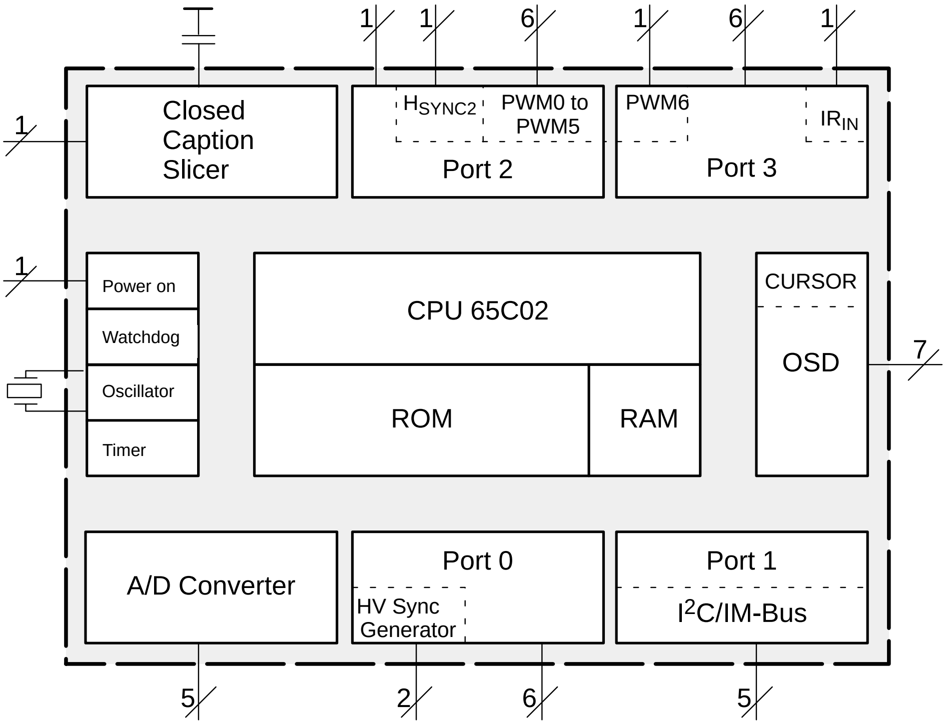


Fig. 1–1: Block Diagram of the CCZ 3005K

2. Functional Description

2.1. CPU

The CPU is a standard 65C02 core.

2.2. ROM

The CCZ 3005K has 62 kBytes of mask-programmable ROM on chip. It covers the addresses from 0800H to FFFFH.

2.3. RAM

1536 Bytes RAM are integrated as two portions:

Table 2–1: RAM configurations

Page	Start	Stop	Length in Bytes
0 and 1	0000H	01FFH	512
3 – 6	0300H	06FFH	1024

Page 0 offers particularly fast access for the CPU and is therefore very valuable for fast, compact programs. Page 1 contains the stack area. Page 3 and following are used as display memory for CCD (Closed Caption Decoding) and OSD (On-Screen Display). Page 2 is reserved as I/O page (the 65C02 has memory-mapped I/O).

2.4. Clock Generator

An integrated two-pin oscillator, accompanied by a programmable divider, generates the clock for the microcontroller. The divider is expressed by the equation

$$f_{system} = f_{XTAL} \ / \ 2*(n+1)$$

where n is a value from 0 to 255. After reset n is set to 0. f_{system} can be modified by writing a new ‘n’ value to address 200H.

Important: **Any kind of display mode (OSD or CCD) requires a system speed of $f_{XTAL/2}$ (n = 0). All timings in the CCZ are based on f_{system}. Other timings than $f_{system} = f_{XTAL/2}$ are not recommended.**

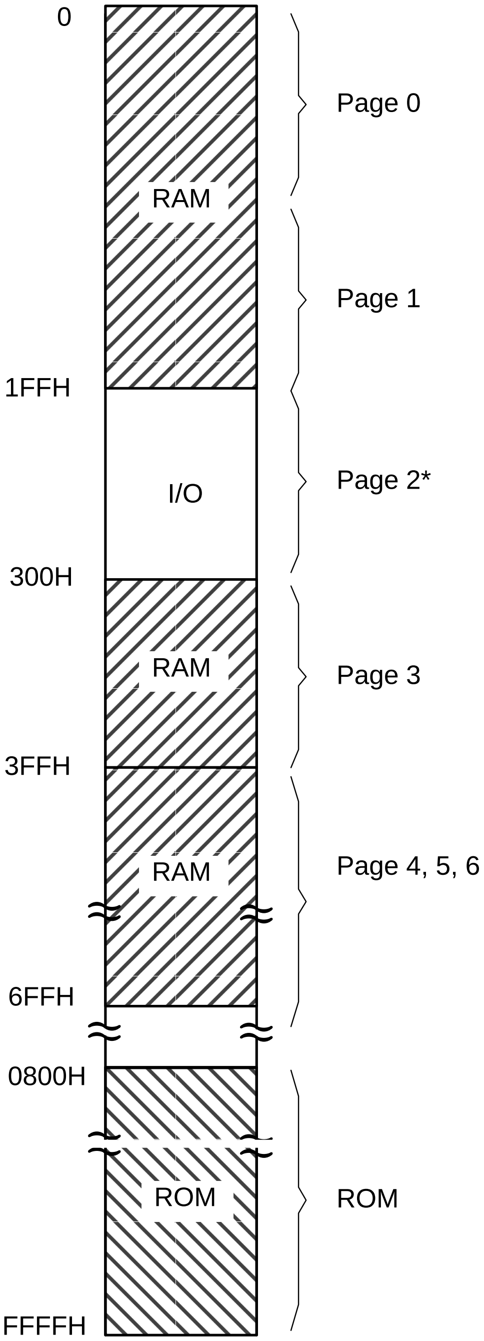


Fig. 2–1: Address map

* With the exception of addr. 02E0H to 02E7H all register addresses of page 2 are internal. With 02E0H to 02E7H external hardware access is possible. These addresses are used for emulation purposes.

2.5. Control Register

This is a combination of control switches in an 8-bit register. During reset it is loaded with the contents of the address FFF9H, but it can also be read and written via software (address 0201H). The switches have the following functions:

bit 0	CPU: ‘0’ = disabled, ‘1’ = enabled
bit 1	RAM; ‘0’ = disabled, ‘1’ = enabled[1]
bit 2	ROM: ‘0’ = disabled, ‘1’ = enabled
bit 3 to 7	set them to 1

[1] To use the emulator chip version:
Set bit 4 to ‘0’. This enables the additional address and data lines. If bit 1 of the control register is set to ‘1’, addresses 0 to 1FFH and 300H to 7FFH are assumed to be inside the CCZ emulator chip. Thus the data bus may not access external devices (RAM) located in this address range. With the control byte “%11101001” = “E9H” the emulator chip can access almost 64 kBytes of external memory. Only the addresses for the internal I/O registers stay internal (page 2 without 2E0H to 2E7H).

The logical level at the TEST-pin during RESET decides whether the control byte is read from internal or external memory. For operation without external memory test pin = low-level is used, with the (internal) control byte set to FFH.

2.6. Reset Function

The internal reset provides a correct basic setup of the complete hardware on the chip. An internal control register (adr. 201H) is loaded during reset with the byte out of address FFF9H. The internal voltage supervision resets the IC if the voltage is too low. If the frequency is too low, the same function is effected by the clock supervision. Once activated and not refreshed correctly, the watchdog also generates a reset (see chapter 2.7. for details). These internal reset sources (watchdog, voltage detector and clock supervision) use the reset pin as output. Internal resistors limit the maximum current.

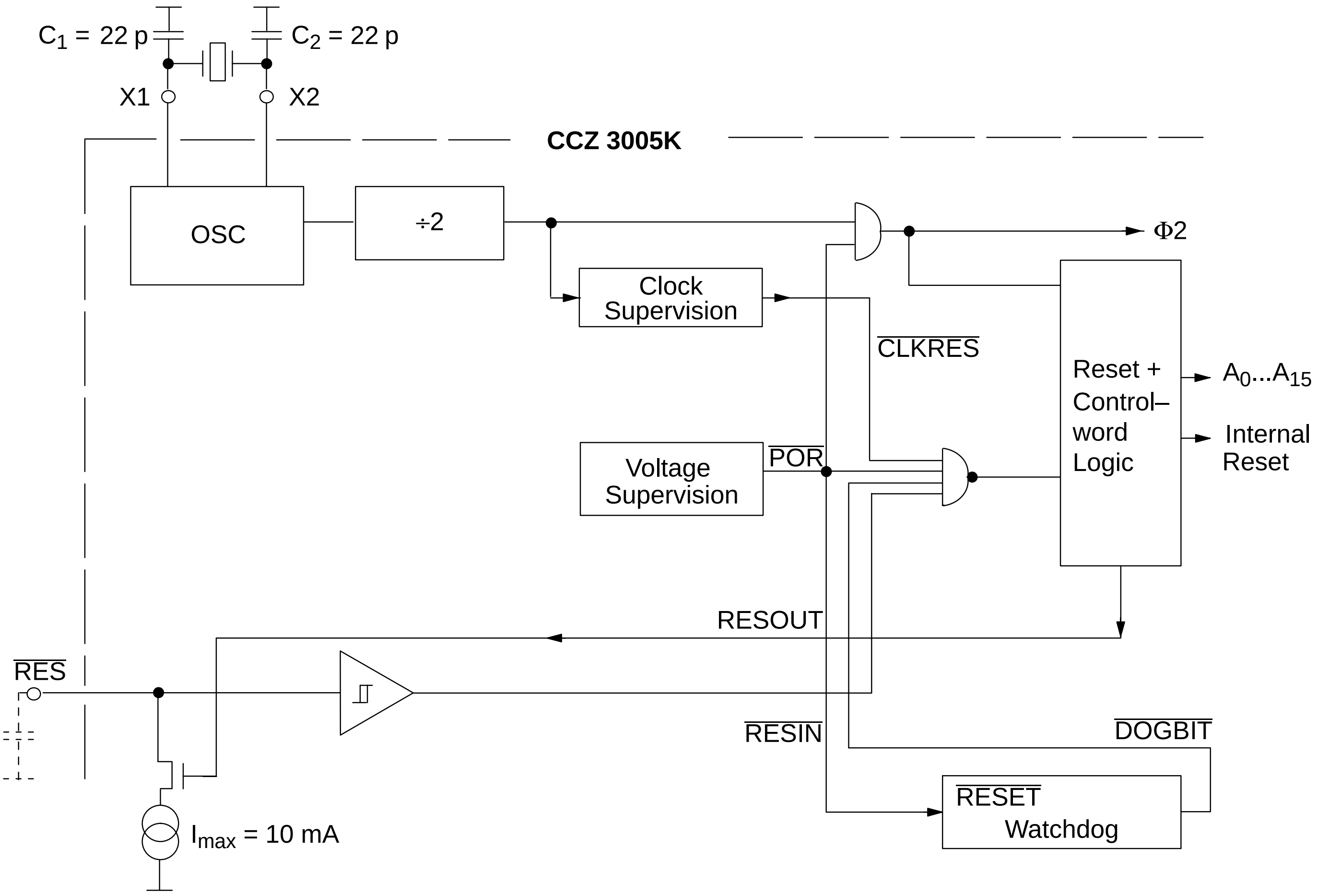


Fig. 2–2: Oscillator and reset

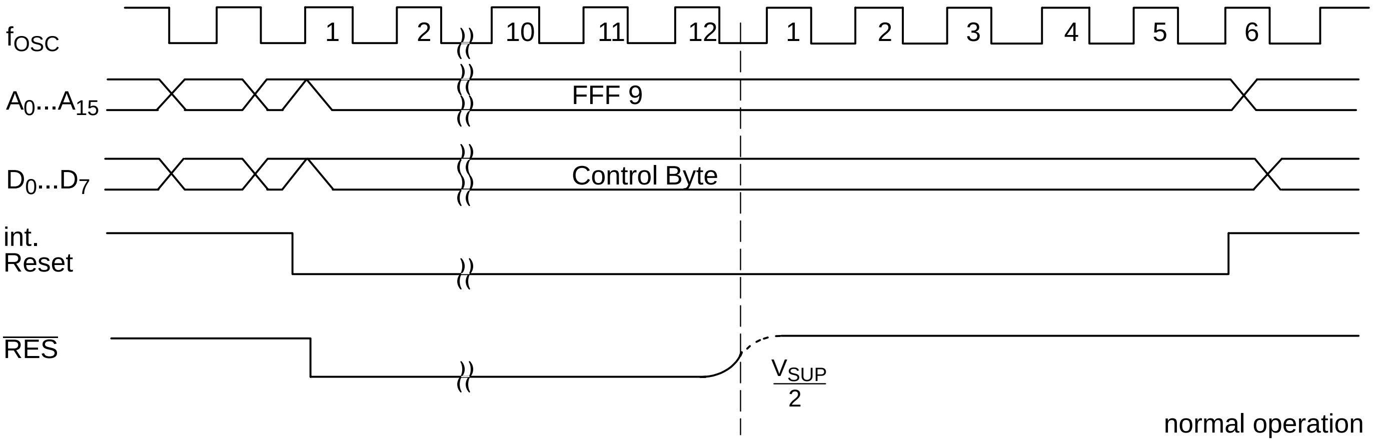


Fig. 2–3: External reset sequence

2.7. Watchdog

This counter circuit offers hardware support for software problems. It is disabled after reset and enabled with the first write of the desired time value into its register. The value to program is calculated by:

(1) $n = (T_{WD} * f_{system} / 65536) - 1$

with n = watchdog counter value to be programmed for T_{WD} = the desired watchdog time and f_{system} = system frequency.

Remarks:

a) To prevent the generation of a “RESET” by the watchdog before it could be retriggered by the software, watchdog counter values of less than 2 should not be programmed.

b) The system clock as input of the watchdog counter is influenced by the system clock prescaler, determining the CPU speed (register addr. 200H).

The software cannot stop this counter, but has to retrigger it by writing the inverted value (one’s complement) of the preceding written pattern into its register, which makes unwanted retrigger loops of disturbed software unlikely. These writes have to occur within the time frame (32 ms to 2 s at 6 MHz system clock), defined with the first write. If no write with the expected pattern occurs within the programmed time period, the watchdog circuit resets the CCZ at the end of the time period.

The software can detect if a reset was generated by the watchdog: bit 0 of the watchdog register is ‘0’ if the last reset was generated by the watchdog. This bit is preset (set to ’1’) only by power-on or writing to the watchdog register. Thus checking it has to occur before the first watchdog register write access.

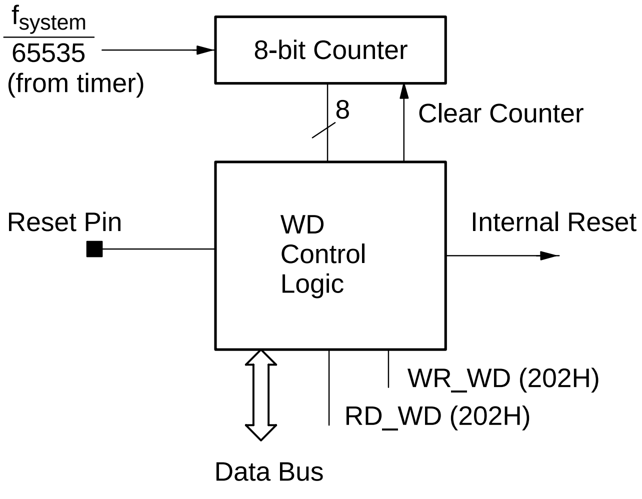


Fig. 2–4: Watchdog

Examples:

To set a cycle time of 1 s with a 6MHz system clock, the value is 91. This value is calculated as follows:

```
system frequency:       6MHz,
watchdog cycle time:    65536 / 6MHz = 10.92ms,
counter value:          1s/10.92ms = 91.55.
```

The nearest integer value is 92. As a 0 loaded into the counter divides by 1, already, the watchdog counter has to be programmed with `92 – 1 = 91`.
Using the above-mentioned equation (1):

```
n = 91 = 1s * 6MHz/65536 – 1
```

the software sequences in Assembler could look like this:

Definitions:

```
;constants:
WATCHDOG_TIME     EQU   91
;CCZ I/O–address:
watchdog_address  EQU   202H
;variable:
watchdog_value    EQU   30H ;(address of free RAM
                            ; location)
```

Example 1:
During initialization the watchdog is filled with the desired time-value:

```
      LDA   #WATCHDOG_TIME
      STA   watchdog_address
      STA   watchdog_value ; memorize pattern
```

In the main loop of the program the watchdog has to be retriggered cyclically

```
      LDA   watchdog_value
      EOR   #FFH
      STA   watchdog_address ; invert bits
      STA   watchdog_value ; memorize new pattern
```

Example 2:
If an interrupt function occurs cyclically, one value may be programmed in the interrupt service routine, while the other is written in the main loop. So both, the continuity of executing the interrupt service and the main loop are checked.

During initialization the watchdog is filled with the desired time value:

```
      LDA   #WATCHDOG_TIME
      STA   watchdog_value; memorize pattern
```

Sequence in the interrupt function:

```
      LDA   watchdog_value
      CMP   #WATCHDOG_TIME
      BEQ   SKIP_IRQ_WD
;
      STA   watchdog_address
      EOR   #$FF
      STA   watchdog_value
SKIP_IRQ_WD:
```

Sequence in the main loop:

```
      LDA   watchdog_value
      CMP   #WATCHDOG_TIME
      BNE   SKIP_WD
;
      STA   watchdog_address
      EOR   #$FF
      STA   watchdog_value
SKIP_WD:
```

Remark:
It is important to program the watchdog register with the new value **before** this value is memorized in the shadow variable, because this procedure could be interrupted by the interrupt which programs the watchdog with the complementary value.

2.8. Ports P0–0 to P3–7

Up to 29 port lines grouped in 4 ports (3 * 8bit, 1*5bit) are available:

P0–0 to P0–7	8 bits
P1–0 to P1–4	5 bits
P2–0 to P2–7	8 bits
P3–0 to P3–7	8 bits

Some of the port lines can be moved into the 'special mode'. Three registers in the I/O-page belong to each port:

– *mode register* write only
(defines each line as port
or special mode pin)
'0' = port mode = reset value

– *tristate register* write only
(disables or enables the
port output stage for each line)
'1' = tristate = reset value

– *data register* read/write
(reads pin levels or writes
port data)
'0' = reset value

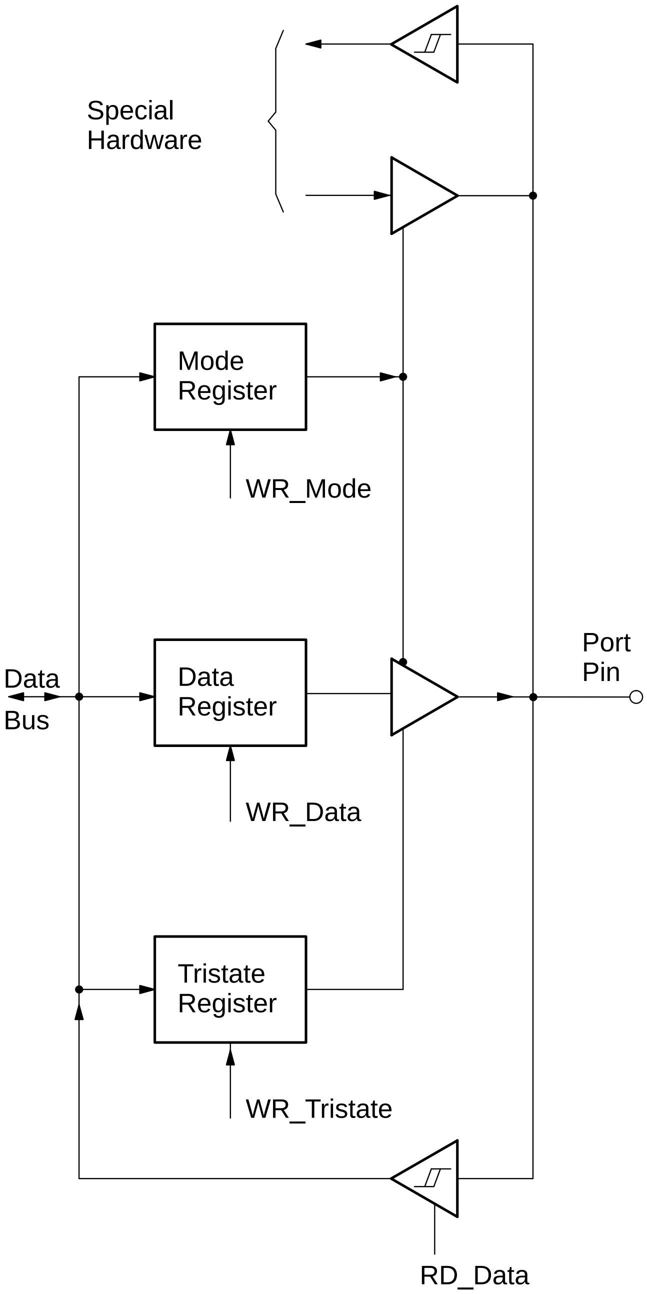


Fig. 2–5: Port logic

After Reset all Ports are in the Port and the output drivers in the tristate mode. The port output drivers have push-pull characteristics. This may be different in the special modes (see description of special mode blocks).

2.9. 6–bit DACs PWM0 to PWM5

Six digital-to-analog converters belong to the CCZ 3005K. The push-pull outputs of the 6-bit PWM-converters are active if in the corresponding port registers the special mode flag is set *and* the tristate flag is reset (output=conducting):

Table 2–2: 6–bit DAC ports

DAC	Port Pin	Data Register Address
PWM0	P20	250H
PWM1	P21	251H
PWM2	P22	252H
PWM3	P23	253H
PWM4	P24	254H
PWM5	P25	255H

By writing a 6-bit value to the converter's data register (D0 to D5 = value, D6, D7 = 0) the software can control the DACs. The minimum position (00H) generates a constant low output signal, the max. value (3FH) a 1/64 low signal. The clock of the PWM-converters is 1/8th of the system clock.

2.10. 14–bit DAC PWM6

The CCZ 3005K is equipped with one 14-bit digital-to-analog converter for tuning voltage synthesis. The push-pull output of the 14-bit PWM-converter is active if in the corresponding port register the special mode flag is set *and* the tristate flag is reset (output=conducting):

Table 2–3: 14–bit DAC port

DAC	Port Pin	Data Register Address
PWM6	P37	256H (6 LSBs) 257H (8 MSBs)

By writing a 14-bit value to the converter's 2 data registers the software can control the 14–bit DAC. The minimum position (0000H) generates a constant low output signal, the max. value (FF3FH) a 1/16384 low signal. Writing into the MSB register (257H) starts the transfer of the complete 14–bit value into the internal PWM6 logic. This register should therefore be written after the LSB register (256H). The clock of the PWM6 converter is the system clock divided by 2.

The PWM6 output needs external circuitry to generate a stable tuning voltage. Fig. 2–6 shows the necessary components.

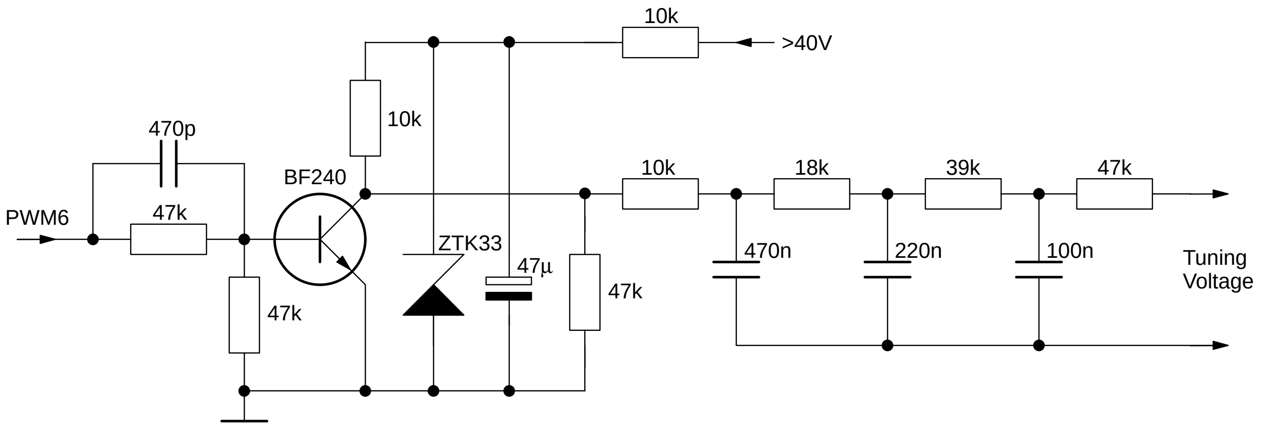


Fig. 2–6: Application circuit to generate tuning voltage

2.11. I^{2}C and IM-Bus Interface

In special mode (conducting), port 1 works as a master bus interface. It can generate two different kinds of format:

– I^{2}C format

– IM format

Two terminals are available: 3 pins (special mode of P12 to P14) as IM or I^{2}C lines and 2 pins (special mode of P10, P11) as I^{2}C. Terminal 2 can only operate as I^{2}C interface because of the missing third line. The MSBit of the bus prescaler registers (address 2DBH) is used to switch between terminals. The remaining 7 bits can be used to set the bit rate.

bit 7	0= terminal 1, 1= terminal 2
bit 6 to 0	bit rate=f_{system}/(4 * n)

where n is the value of bits 0 to 6 and the setting value (0 = reset state means n = 128). A complete telegram is assembled by the software out of individual sections. Each section contains an 8-bit data. This data is written into one of the nine possible Control-Data registers. Depending on the chosen address, a certain part of an I^{2}C or IM-bus cycle is generated. By means of corresponding calling sequences it is therefore possible to join even very long telegrams (e.g. long data files for auto increment addressing of I^{2}C slaves).

The software interface contains a 3 word deep FIFO for the control-data registers as well as for the received data. Thus all IM and most of the I^{2}C telegrams can be transmitted to the hardware without the software having to wait for empty space in the FIFO.

All address and data fields appearing on the bus are constantly read and written into the Read-FIFO. The software can then check these data in comparison with the scheduled data. If a read instruction is handled, the interface must set the data word FFH so that the responding slave can insert its data. In this case the Read-FIFO contains the read-in data.

If telegrams longer than 3 bytes (1 address, 2 data bytes) are received, the software must check the filling condition of the control data FIFO and, if necessary, fill it up (or read out the Read-FIFO). A variety of status flags is available for this purpose.

Moreover, in the I^{2}C mode the ACK-bit is recorded separately on the bus lines for the address and the data fields. However, the interface itself can set the address ACK=0. In any case the two ACK flags show the actual bus condition. These flags remain until the next I^{2}C start condition is generated.

To minimize disturbances generated by the I^{2}C signals, the fall times of both the I^{2}C-SCL and I^{2}C-SDA outputs on both I^{2}C terminals are increased to one f_{SYSTEM} cycle while its output currents are decreased to one third of its maximum. Thus by switching from high to low, it takes one f_{SYSTEM} cycle until the maximum driver current is switched on. It depends on the sizes of the (external) load capacitance and pull-up resistor when the low level is reached (see Fig. 2–8). This feature is only active when the port output buffers are current-controlled, i.e., bit 5 of the “Hardware Control Register”, address 209H, is set to ‘1’.

Capacities on any of the I^{2}C pins should not exceed 100 pF. Bigger capacitors could effect higher disturbances. R_{LOAD} should be equal to or greater than 2 kΩ.

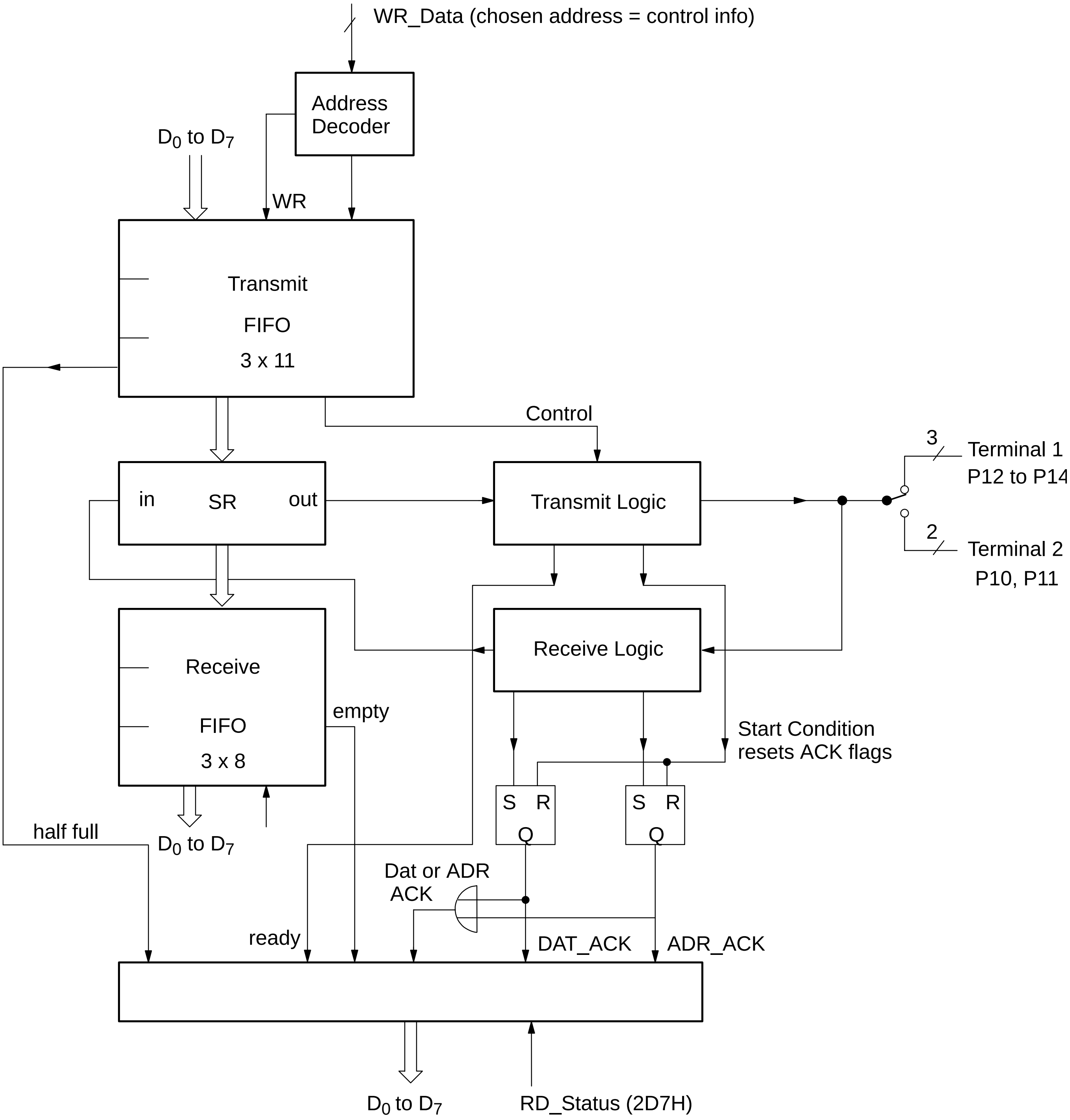


Fig. 2–7: I^{2}C/IM-Bus Interface

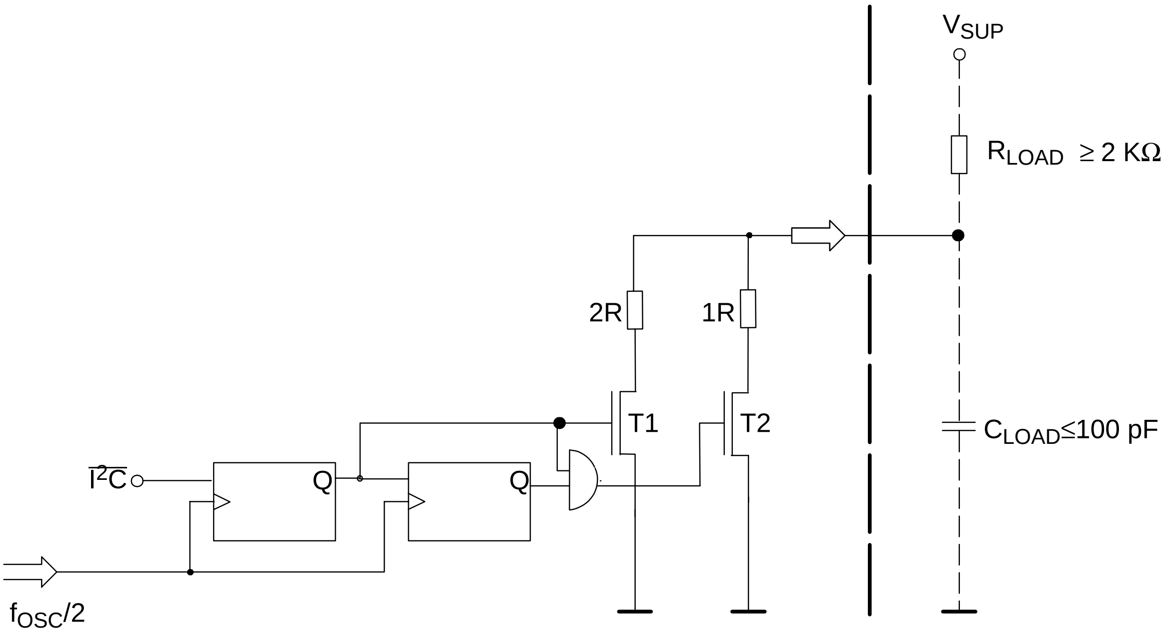


Fig. 2–8: Principle of I^2C signal outputs current reduction, during first T_{OSC}, after switching from high to low

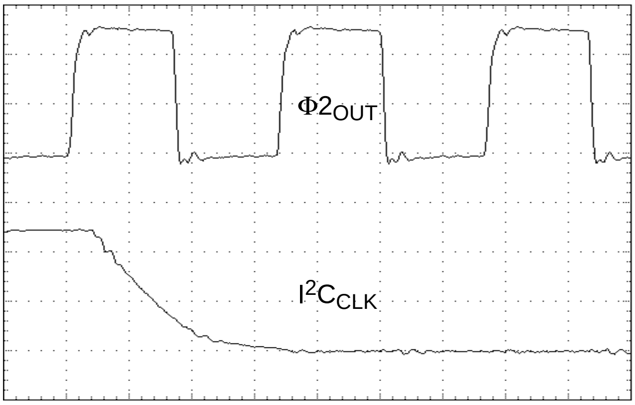


Fig. 2–9: Current reduction active, bit 5 of 'Hardware Control Register' set to '1'
R_{LOAD} = 2 KΩ, C_{LOAD} = 100 pF, recommended

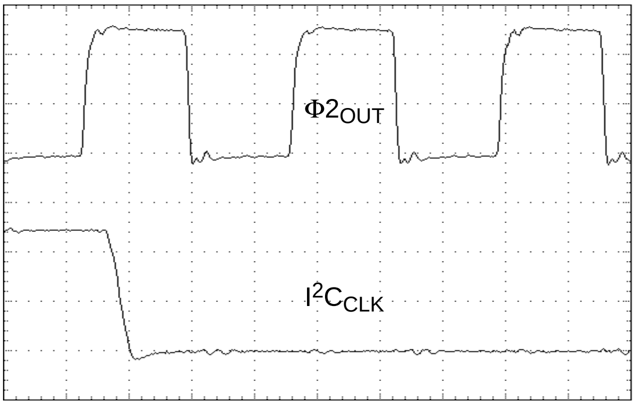


Fig. 2–11: Current reduction inactive, bit 5 of 'Hardware Control Register' set to '0',
R_{LOAD} = 2 KΩ, C_{LOAD} = 100 pF

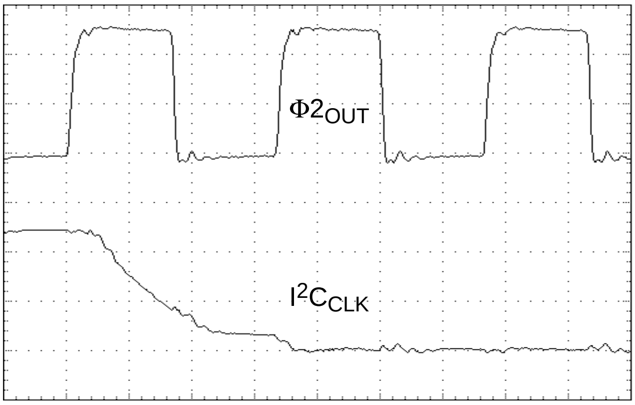


Fig. 2–10: Current reduction active, bit 5 of 'Hardware Control Register' set to '1', R_{LOAD} = 1 KΩ, C_{LOAD} = 100 pF, not recommended

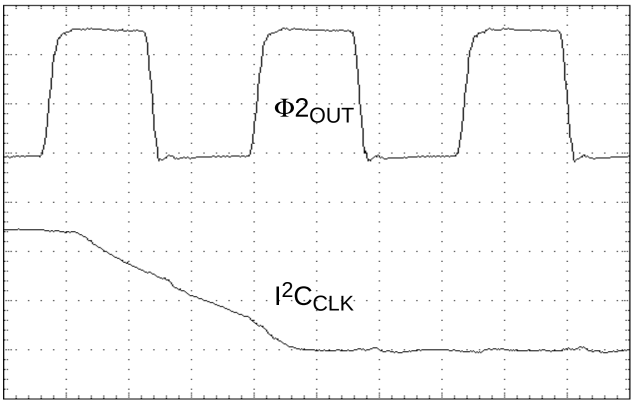


Fig. 2–12: Current reduction active, bit 5 of 'Hardware Control Register' set to '1' ,
R_{LOAD} = 2 KΩ, C_{LOAD} = 350 pF, not recommended

Table 2–4: I^{2}C and IM-bus interface registers

Address	Function	
2D0H(w)	generates I^{2}C start condition, transfers Data as I^{2}C address and sets address ACK=1	
2D1H(w)	same as above, ACK=0	
2D2H(w)	output 8 I^{2}C Data bits, set ACK=1	
2D3H(w)	same as above, set ACK=0	
2D4H(w)	output 8 I^{2}C Data bits, sets ACK=1, generates I^{2}C stop condition	
2D5H(w)	same as above sets ACK=0	
2D6H(r)	receives FIFO	
2D7H(r)	status flags:	
	bit 0	not used
	bit 1	1= receive FIFO empty
	bit 2	1= contr-data-FIFO half full
	bit 3	1= Bus busy
	bit 4	I^{2}C data ACK
	bit 5	I^{2}C adr ACK
	bit 6	“OR”ed ACK
	bit 7	not used
2D8H(w)	generates IM-address field	
2D9H(w)	generates 8 IM-data bits	
2DAH(w)	generates 8 IM-data bits and the IM-stop condition	
2DBH(w)	terminal select & speed	

For example, the software has to work off the following sequence (ACK = 1) to read a 16-bit word from an I^{2}C device address 10H (on condition that the bus is not active):

–write 21H to 2D0H
–write 0FFH to 2D2H
–write 0FFH to 2D4H
–read dev. address 2D6H
–read 1st databyte 2D6H
–read 2nd databyte 2D6H

$\left.\begin{matrix}\text{–read dev. address 2D6H}\\ \text{–read 1st databyte 2D6H}\\ \text{–read 2nd databyte 2D6H}\end{matrix}\right\}$ check receive FIFO empty flag (bit 1, 2D7H) before read

The value 21H in the first step results from the device address in the 7 MSBs and the R/W-bit (read=1) in the LSB. If the telegrams are longer, the software has to ensure that neither the Control-Data-FIFO nor the Read-FIFO can overflow.

To write data to this device:
–write 20H to 2D0H
–write 1st data byte to 2D2H
–write 2nd data byte to 2D4H

The bus activity starts immediately after the first write to the Control-Data-FIFO. In the I^{2}C mode the transmission can be synchronized by an artificial extension of the Low phase of the clock line. Transmission is not continued until the state of the clock line is High once again. Thus a slave (software slave!) can adjust the transmission rate to its own abilities.

The I^{2}C/IM-bus interface is a pure Master system, Multimaster busses are not realizable.
The clock and data terminal pins have open-drain outputs. The IM-Bus-Ident Line (terminal 1 only) is a push-pull output stage (see chapter 3.6.5. on page 54).

UNIT = Bit-Period

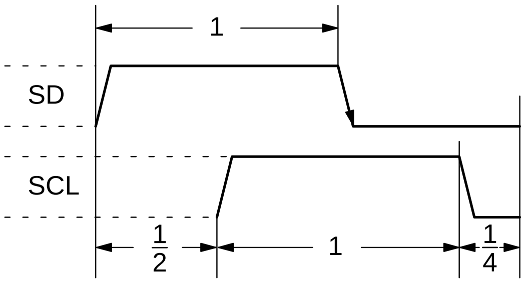


Fig. 2–13: Start condition I^{2}C-bus

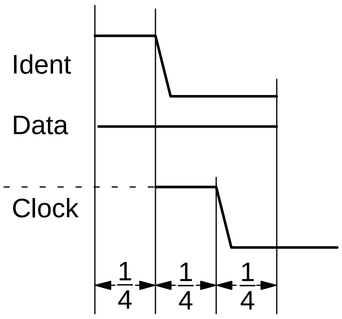


Fig. 2–16: IM-bus start condition

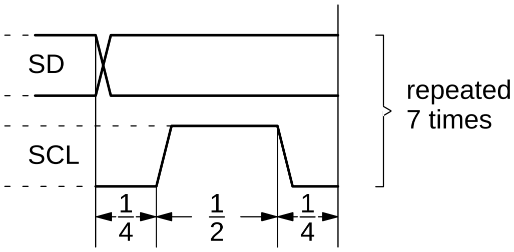


Fig. 2–14: Single bit on I^{2}C-bus

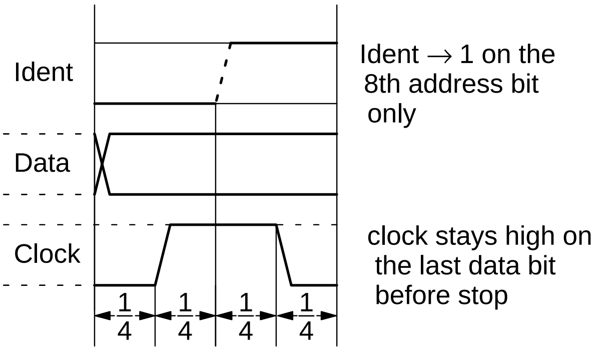


Fig. 2–17: Single bit on IM-bus

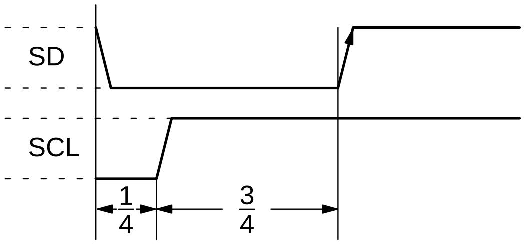


Fig. 2–15: Stop condition I^{2}C-bus

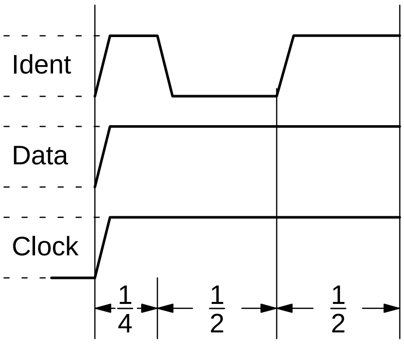


Fig. 2–18: Stop condition IM-bus

2.12. A/D Converter

The analog voltages at 5 pins of the CCZ can be converted to an 8-bit digital value. With an input multiplexer one of these five inputs is selected. The input voltage is ‘sampled and held’ during conversion time. Conversion gets started with writing the number of the desired analog input pin into the ‘Analog Input Select and Status’ register (address 2A8H). After waiting until the ‘End of Conversion’ (‘EOC’) flag in this register (bit 7) is set to ‘1’, the result is available in the ‘A/D Converter Output’ value register (address 2A9H).
The byte representing the analog voltage at the chosen ADC input pin is evaluated by

digital value = integer value of (256 x V_{IA}/V_{SUPA}).
V_{IA} = Analog Input Voltage
V_{SUPA} = Analog Supply Voltage

The result of this equation is valid for $GND_A < V_{IA} < V_{SUPA}$. For $V_{IA} \geq V_{SUPA}$ the digital value is its maximum: FFH.

The converter needs 68 oscillator clock cycles to sample an input, so with a 12 MHz crystal it takes 5.67 µs to convert an analog value. During this time the input voltage should be kept as stable as possible (see Fig. 2–19).

The ‘EOC’ flag is comparable with a busy signal and only readable. It is reset (set to ‘0’) by writing into the ‘Analog Input Select’ register and choosing one of the 5 analog inputs that starts conversion. A CCZ Reset also resets the ‘EOC’ flag and selects the ADC0 input pin.

Analog Input Select and Status Register (Address 2A8H):

Bit 7:	EOC-flag (read only)
Bits 2 to 0:	Number of analog input pin (0 to 4) (write only)

Important:

The ADC hardware must be enabled before the first conversion after RESET can be started, by setting bit 3 of the ‘Hardware Control’ register (addr. 209H) to ‘1’. Then it is necessary to wait for 68 oscillator clock cycles before the first conversion can be started, by writing the desired ADC input number into the ‘Analog Input Select’ register. Any conversion started earlier delivers useless results.

Application Tips:

The input capacitance of an analog input is about 23 pF. The maximum input current depends on the voltage step at the input capacitance and has to be considered when calculating the signal voltage divider. An external capacity between the ADC pin and GND should have a value of at least 10 nF. V_{IA} should be as stable as possible during sample time.

V_{SUPA} must not exceed V_{SUP}!

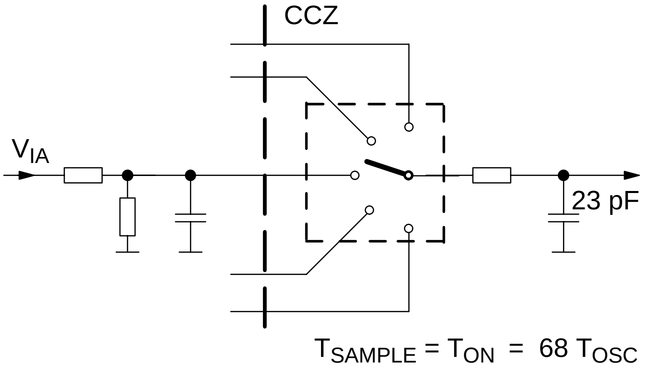


Fig. 2–19: A/D converter input

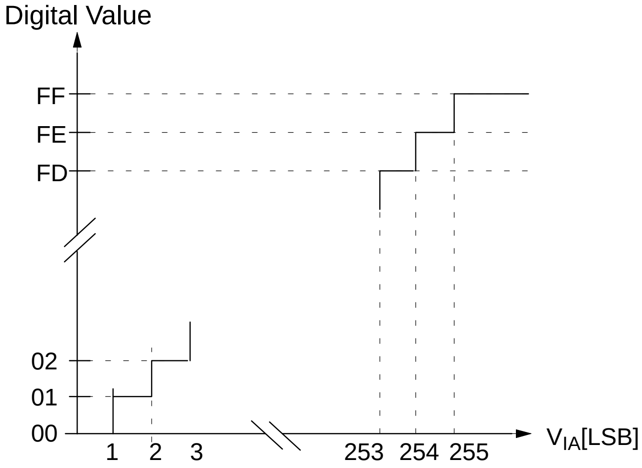


Fig. 2–20: A/D converter diagram

2.13. Closed Caption Acquisition

2.13.1. Video Input

The AC-coupled composite video input signal is applied to the VIDEO IN pin where the negative horizontal sync tip is clamped to a value of 1.911V DC via A1. The gated sync tip clamp circuit is designed to work with a 0.1 µF coupling capacitor. The run-in clock of the Closed Caption signal is averaged by R1 and C1. C1 is an external capacitor with a value of 560 pF. The resulting average level is taken as the optimum level for slicing the digital data which follows the run-in clock. It is important for C1 to hold the slicing level for a whole frame (33.2 ms). The best slice level, in terms of common mode range for the chip, is one half of the minimum supply voltage for the CCZ: 4.75V : 2= 2.375V. Since a 1 Vpp video input signal is nominal the clamp level should set the 25 IRE point (data slice level) for the 1 Vpp case to 2.375V. This is done by clamping the sync tips to 1.911 V (2.375 V–0.464 V).

Table 2–5: Video Input Levels, Clamped, in Volts

Video Signal Input Level [IRE]	Clamp Level at Video Signal Input 1.0 V_{pp} [V]	Clamp Level at Video Signal Input 2.0 V_{pp} [V]	Remarks
100	2.911	3.911	(white)
50	2.554	3.196	–
25	2.375	2.839	(slice)
7.5	2.250	2.589	(black)
0	2.196	2.482	(back porch)
–40	1.911	1.911	(sync)

The data slicer consists of an on-chip low-pass filter and a high-speed comparator. The sliced data passes through a glitch filter which ignores spikes of less than three f_{XTAL} (12 MHz) periods in duration. The data is then conditioned in a preprocessor stage and transferred to the CPU. The software reads the data from the selected caption line(s) and decodes the caption data further.

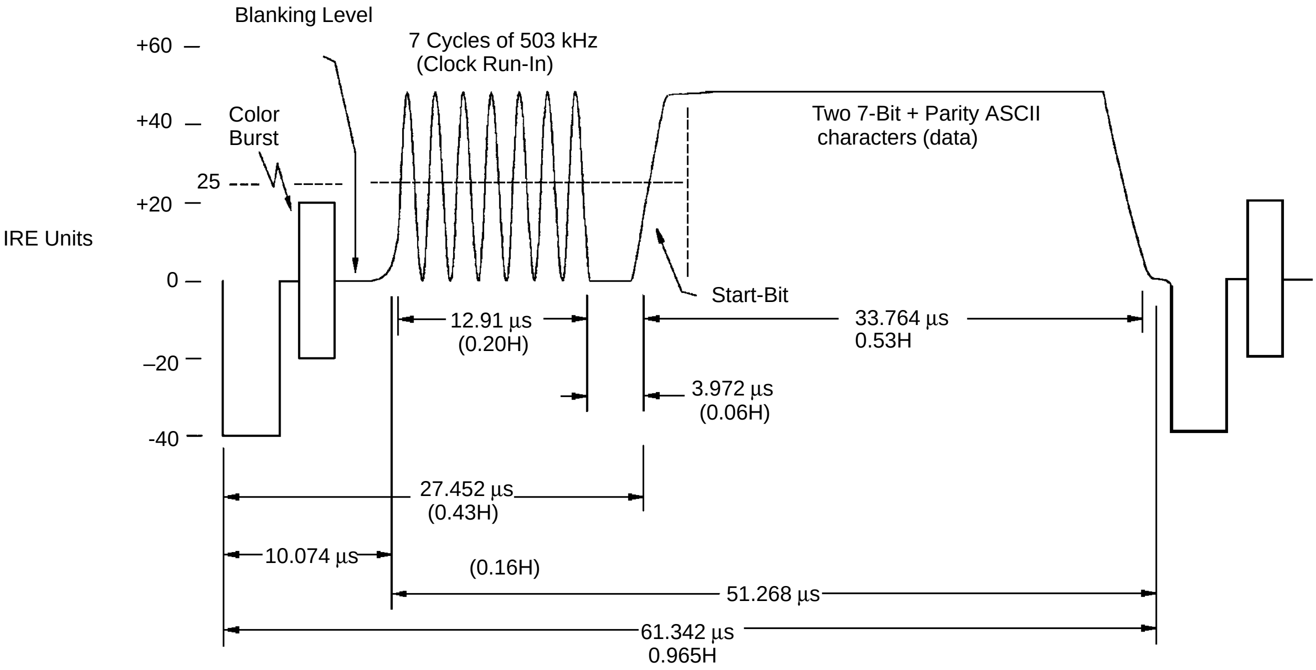


Fig. 2–21: Line 21 Field 1 Data Signal Format

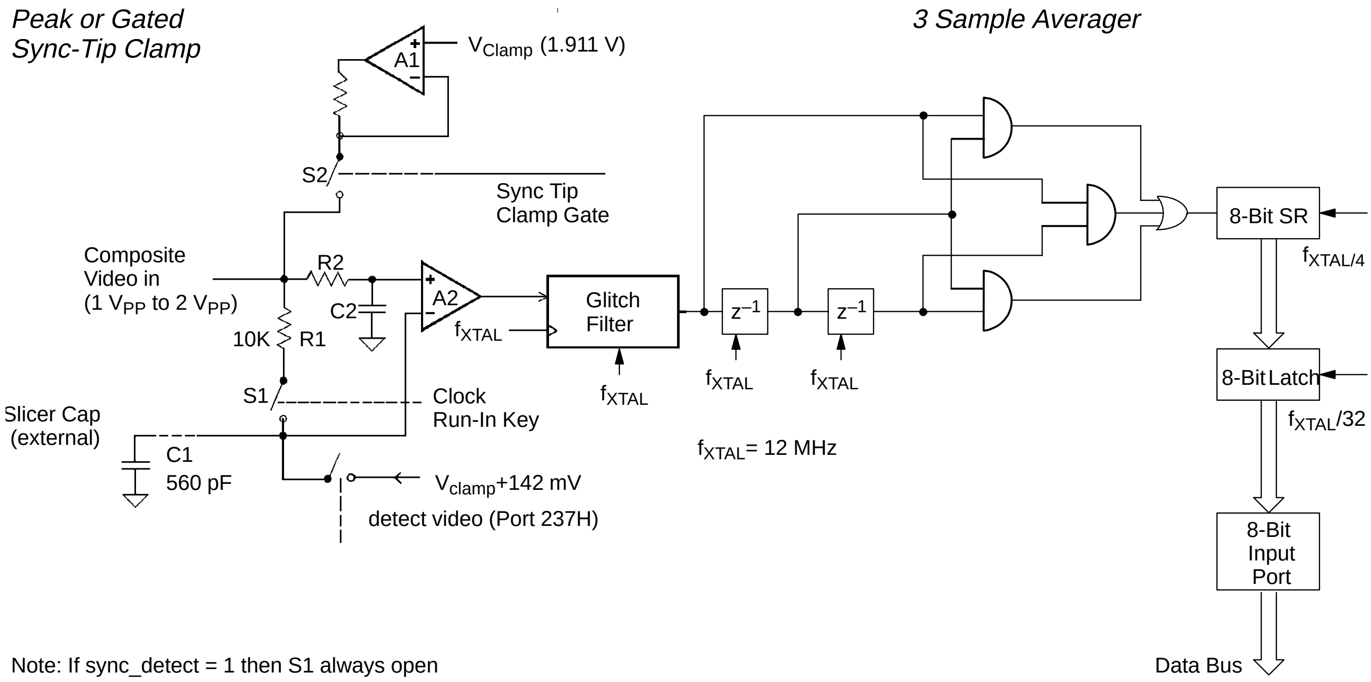


Fig. 2–22: Principle of video input for the detection of Closed Caption Data

2.13.2. Closed Caption Data Detection

The sliced data are shifted serially into an 8-bit register with the $f_{XTAL}/4$ as clock signal. The $f_{XTAL}/32$ latches these 8 bits so that, as byte, they are available for the processor at register 23FH. As every $f_{XTAL}/32$ period overwrites the previous latched data, the CPU has to read this register fast enough to get all data. The software part doing so could look like this:

```
;Capture data port address:
capture_data_ EQU $23F

; variables:
capt_buffer_EQU      $50              ;     26 bytes caption data buffer
no_of_data_EQU       capt_buffer_+26  ;     data counter

CAPTURE    LDX       #0               ;     init pointer to captured data in RAM
           LDA       #26              ;     acquire 26 samples
           STA       no_of_data_
CAPTURE_   LOOP
           LDA       capture_data_    ;     read sliced data               ;   4 cycles
           STA       capt_buffer_,X   ;     store it in RAM                ;   4 cycles
           INX                        ;     point to next location in RAM  ;   2 cycles
           CPX       no_of_data_      ;     done 26 samples yet?           ;   3 cycles
           BNE       CAPTURE_LOOP     ;     no, so keep looping            ;   3 cycles
           ;                                                                 ---------
           ;                                                                 16 cycles
```

It is important to have all variables used here defined in the address space of the zero page, and not to cross a page boundary with the loop, as otherwise the execution time increases. With 16 processor cycles the sliced data scan rate of this sequence is 12 MHz/2/16 = 375 kHz at a crystal frequency of 12 MHz. This is exactly the clock of the 8-bit slice data latch of f_{XTAL}/32. Caption data are now in the CCZ memory and have to be decoded. The caption data rate is 503 kHz. The shift rate of the slice data register is f_{XTAL}/4 = 12 MHz/4 = 3 MHz. This is almost 6 times the caption data rate of 6*503 kHz = 3.018 MHz. Thus, in the received bit stream, a bit of the caption data is represented by 6 successive sample bits of the same value (the levels of the clock run-ins represented by 3 bits). As the caption data rate is higher than the scan rate and the phases of both clocks are not synchronized, it may occur that a caption data bit is detected as 5 sample bits only:

clock-drift = (503 kHz- 500 kHz)/500 kHz = 0.6%.

The software to decode the captured data has to consider this item. The decoding of the data can be done by several methods: either by comparing the bit-pattern for the last clock run-ins, the start condition and the data bits, or by searching the start condition first, the last clock run-ins next, and then the data, by checking one or several adjacent bits in the center of the six bit portions. The following example shows what the bit pattern of captured data in the CCZ memory could look like:

Captured data in memory:

```
capt_buffer_:
FCB 00H, 00H, FEH, E7H, 3DH, 8FH, E3H, 00H
FCB F0H, FFH, C0H, 0FH, 00H, C0H, 0FH, FCH
FCB 00H, 00H, 00H, C0H, FFH, 03H, 00H, 00H
FCB 00H, 00H
```

These data are gathered from the following bit stream (as the reception starts with the least significant bits, the notation starts with them, from left to right):

```
00000000  00000000  01111111  11100111
10111100  11110001  11000111  00000000
00001111  11111111  00000011  11110000
00000000  00000011  11110000  00111111
00000000  00000000  00000000  00000011
11111111  11000000  00000000  00000000
00000000  00000000
```

The start-bits begin in the eighth byte:

```
00000000  00000000  01111111  11100111
10111100  11110001  11000111  00000000
                              <start
sequence      >
00001111  11111111  00000011  11110000
00000000  00000011  11110000  00111111
```

In the 6th and 7th byte the last clock run-ins may be detected:

```
00000000  00000000  01111111  11100111
10111100  11110001  11000111  00000000
           <   clock run-in     >
```

The data start in the 10th byte:

```
00000000  00000000  01111111  11100111
10111100  11110001  11000111  00000000
00001111  11111111  00000011  11110000
            <‘1’>   <‘0’>  <‘1’>   <‘0’
00000000  00000011  11110000  00111111
> <‘0’>   <‘0’>   <‘1’> <‘0’>    <‘1’>
00000000  00000000  00000000  00000011
<‘0’>  <‘0’>    <‘0’>  < ‘0’ > <‘0’><
11111111  11000000  00000000  00000000
 ‘1’> <‘1’>
```

The detected bit stream is:

1010001010000011
or
1010 0010 1000 0011
without parity bits (odd parity) and with the most significant bits notices on the left side:

1000101 1000001

i.e.:
1000101B = 45H = ’E’
1000001B = 41H = ’A’

So “EA” is received.

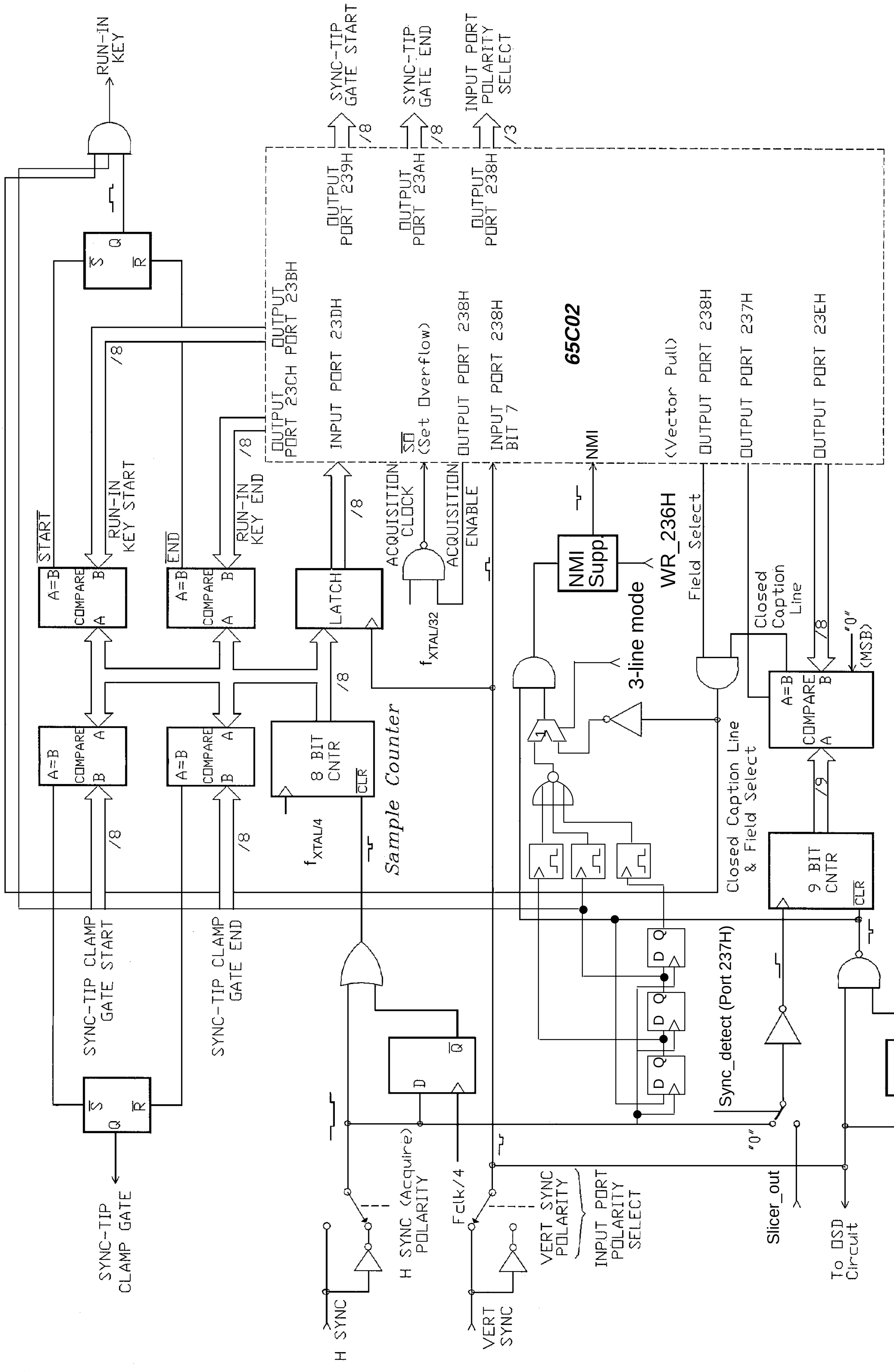


Fig. 2–23: Gate and Window Logic

2.13.3. Gate and Window Logic

The Gate & Window logic is responsible for keeping track of which video field is present at any given time (odd or even field, also known as “Field 1” or “Field 2”), which video line is present (1 to 262 for NTSC, 1 to 312 for PAL/SECAM) and where we are within the line. By keeping track of these three values the Gate & Window logic is able to produce the Run-in-key pulse, which is used by the front-end hardware to determine the best slicing level and to inform the CPU when it is time to start acquiring data from the closed caption video line. The Gate & Window Logic requires a horizontal sync input in addition to the vertical sync input. The active horizontal and vertical pulse width must be at least 6 $T_{OSC,}$ i.e., with a crystal frequency of 6 MHz, it must be greater than or equal to 1 µs.

Vertical Timing

Every TV video *frame* is made up of two video *fields,* an odd field and an even field. The CCZ 3005K must be able to distinguish an odd field from an even field because the telecaption data can appear on either or both fields. The CCZ is able to distinguish between odd and even fields, taking advantage of the fact that there is a half-line offset between fields (since 525 & 625 are odd numbers). The difference is measured and used as a basis for determining which field is which. Note that the position of the Vert Sync and Hor Sync is also a function of the sync processor circuit used to provide Vert and Hor Sync. The phasing of Vert Sync can change from one type of Sync processor to another; but there will always be a measurable difference between the fields. Therefore it is up to the user of the CCZ to determine what phasing is suitable for his particular sync processor.

The Gate & Window logic measures the difference in phasing between Vert Sync and Hor Sync with an 8-bit “Sample” counter (‘Vertical Sync Phase Value’ register, addr. 23DH). This counter is clocked at $f_{XTAL}/4$ (3 MHz) and is cleared by the Hsync signal, either derived from the Hsync pin or the COMPOSITE VIDEO, determined with bit 1 in the ‘Window Logic Control Register2’, (addr. 237H). The active edge of VertSync latches the contents of the counter and generates an interrupt request (NMI). The CPU reads the contents of the latched sample counter and determines whether the new field is odd or even. The logic keeps track of lines with a 9-bit line counter which is clocked each horizontal line and cleared by Vert Sync. The desired closed caption line is provided from a loadable register to a comparator as 8 bits. The 9th bit is hardwired “low”, so the closed caption line must be in the first 256 lines of the field. The closed captioning decode is combined with the “Field Select” line to generate an interrupt to the CPU. The Field line select is provided by the CPU since it knows what field it is trying to find data on. The closed captioning decode also enables the Run-in key.

The acquisition clock is enabled by the CPU at the beginning of the acquisition interrupt routine. The program keeps the acquisition clock enabled for a few cycles beyond the closed caption line to ensure that all of the data has been read from the shift register in the front end. The acquisition clock, $f_{XTAL}/32$, is combined with the closed caption line decode. The CPU–$\overline{SO}$ input is active after RESET.

The CPU needs to know what causes the interrupt (NMI) since the VertSync pulse also generates an NMI. This is accomplished by feeding the Vert Sync pulse directly into a status register. If this bit is high during the interrupt then the CPU assumes this is a Vert Sync interrupt. This leads to the constraint that the Vert Sync signal must remain high for at least one horizontal line period and must be low during the closed caption line. This is normally the case for sync processors anyway. Also, the Vert Sync interrupt routine must be completed before the acquisition interrupt occurs (see Fig. 2–25).

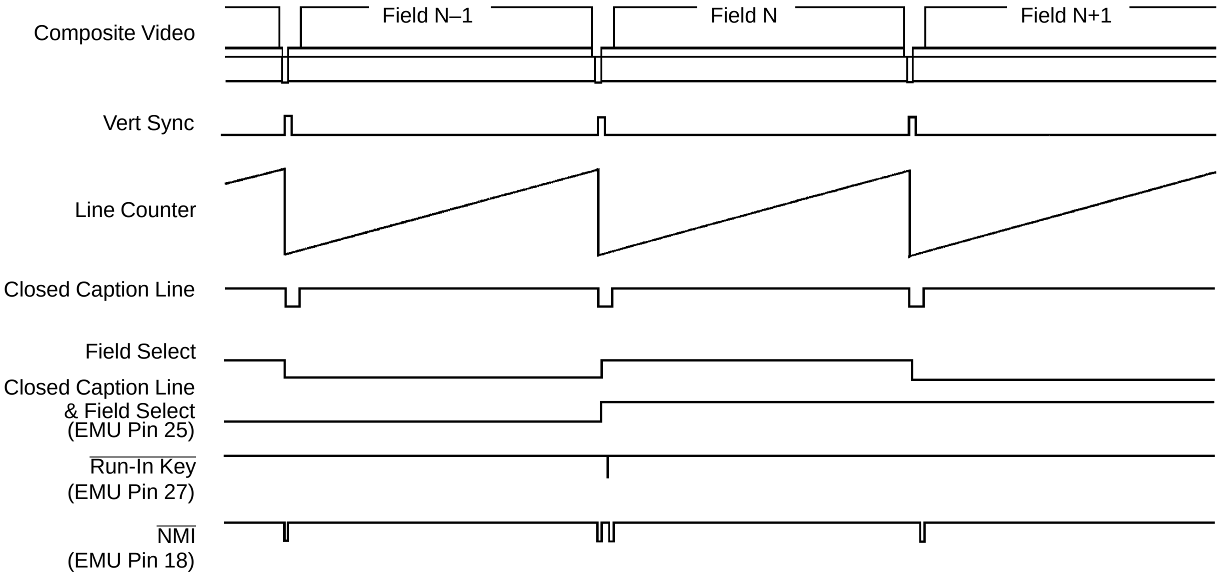

Composite Video
Field N–1
Field N
Field N+1
Vert Sync
Line Counter
Closed Caption Line
Field Select
Closed Caption Line & Field Select (EMU Pin 25)
$\overline{\text{Run-In Key}}$ (EMU Pin 27)
$\overline{\text{NMI}}$ (EMU Pin 18)

Fig. 2–24: Vertical timing

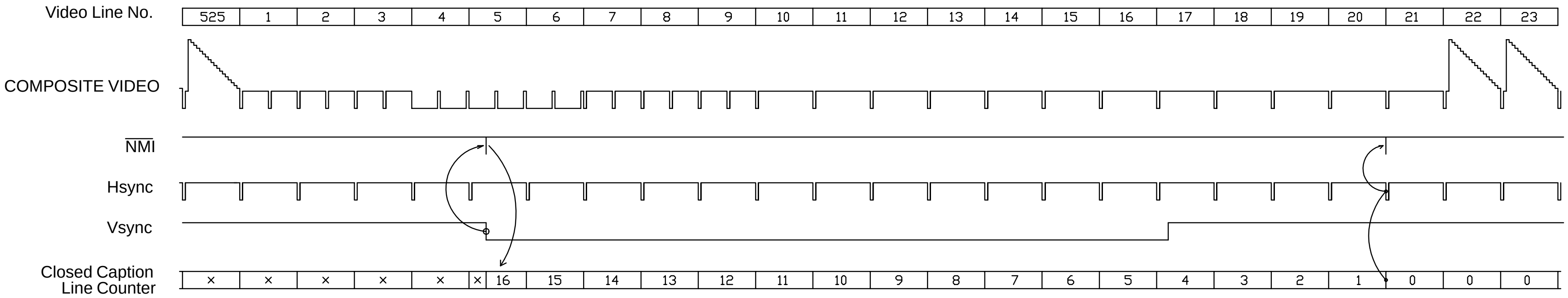


Fig. 2–25: Closed Caption Line Detection

Closed Caption Line Detection:
'Video Detect'-bit (= bit 1 of window logic control register 2, addr. 237H) = '0'
The closed caption line number is defined in the $\overline{\text{NMI}}$ interrupt function.
The line counter is triggered by the Hsync signal.

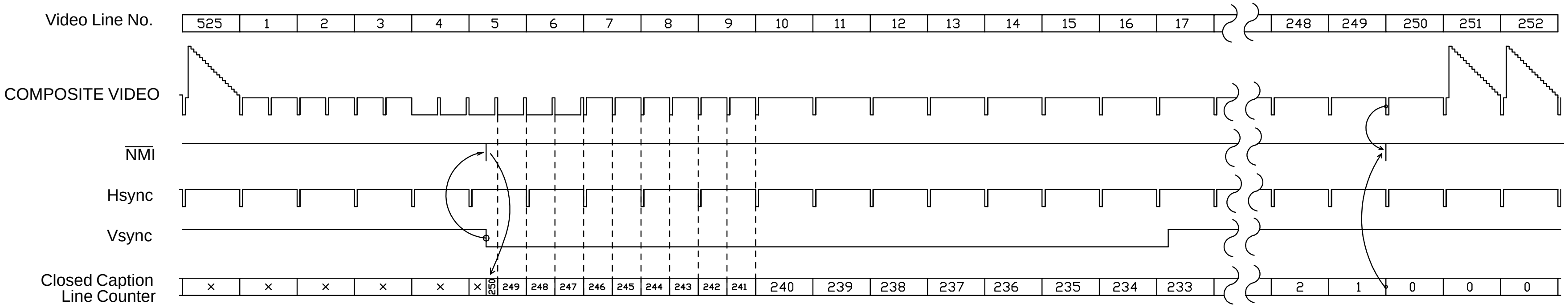


Fig. 2–26: Video Detection

Video Detection:
'Video Detect'-bit (= bit 1 of window logic control register 2, addr. 237H) = '1'
The closed caption line number is defined in the $\overline{\text{NMI}}$ interrupt function.
The line counter is triggered by the Hsync-clamped composite video signal.

Horizontal Timing

The horizontal timing is based on the Hsync input derived from the Hsync_1-pin (default) or the Hsync_2 pin (P27 in special mode), defined with bit 3 in the 'Window Logic Control' Register (addr. 237H). The critical horizontal rate job performed by the CCZ's Gate & Window Logic is to generate the run-in key pulse during the telecaption line. The run-in key signal may be measured on pin 27 of the emulator chip. Its start time is programmable in register 23BH, while its stop time is determined in register 23CH. An optimized timing of the run-in key referred to the composite video signal delivering the caption data on pin 122 of the CCZ 3005K emulator chip is given in figure 2–28.

The key pulse is used by the front-end circuit to establish the optimum data slicing level. The key pulse start and stop points along the horizontal line are decoded from the sample counter. The decodes are programmable. The start/stop points are combined with the "Closed Caption Line" signal to form a Run-in key that occurs only during the closed caption line.

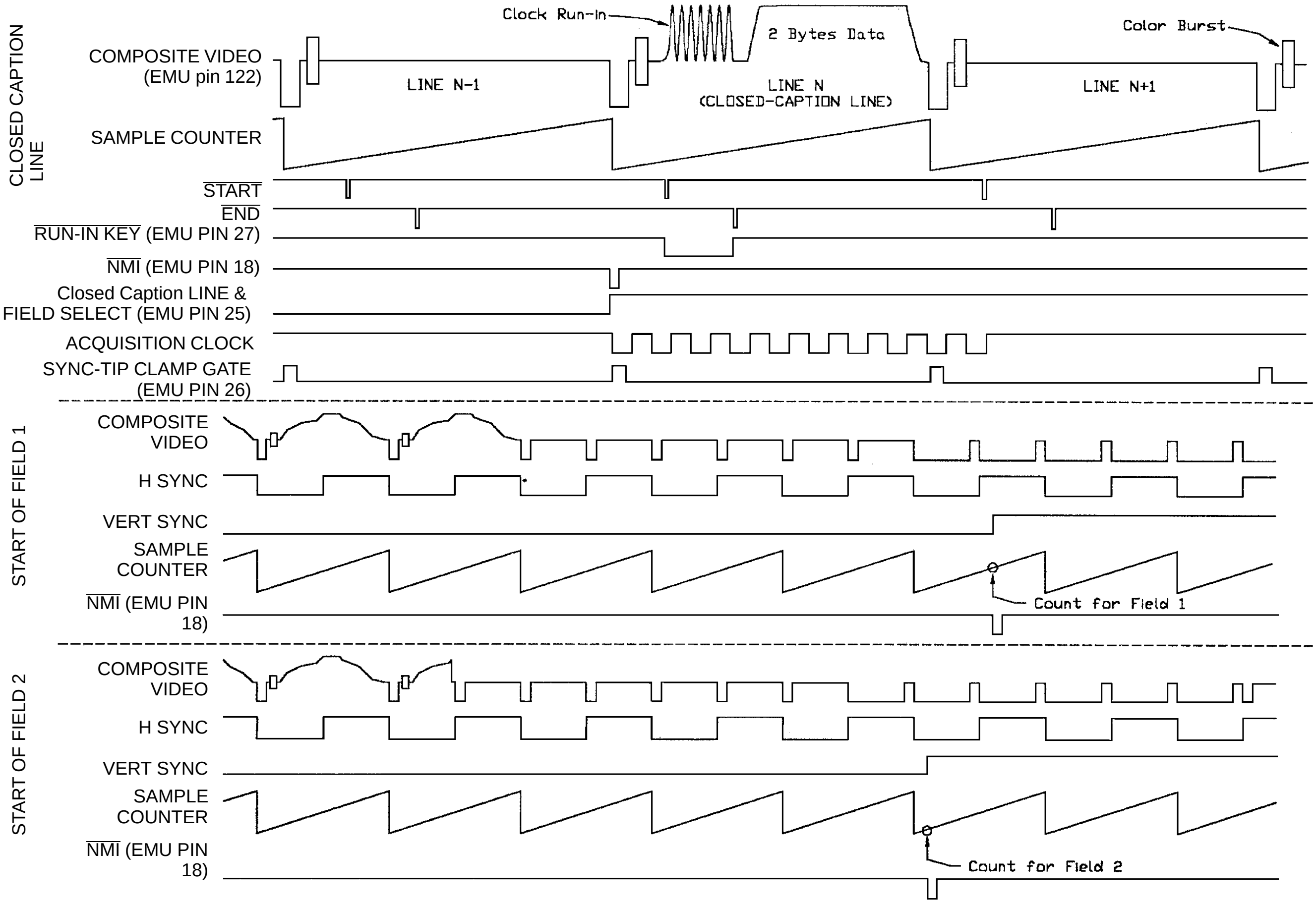


Fig. 2–27: Horizontal timing

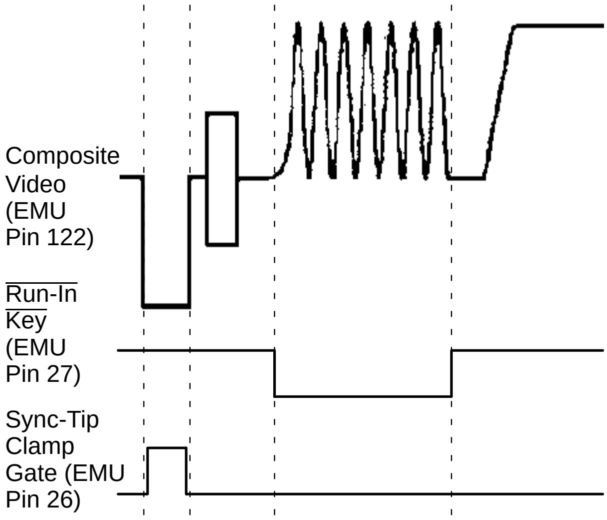

Composite Video (EMU Pin 122)
$\overline{\text{Run-In Key}}$ (EMU Pin 27)
Sync-Tip Clamp Gate (EMU Pin 26)

Fig. 2–28: Optimized Run-In Key and Sync-Tip Clamp

A sync tip clamp gate is also decoded from the sample counter. This gate will start a few counts after the leading edge of Hsync and will end a few counts before the trailing edge of the Hsync portion of the video signal. The start and stop counts are programmable. The sync tip clamp gate signal may be measured on pin 26 of the emulator chip. Its start time is programmable in register 239H, while its stop time is determined in register 23AH. An optimized timing of the sync tip clamp gate referred to the composite video signal delivering the caption data on pin 122 is given in figure 2–28. The sync tip clamp gate is used by the front-end circuit to clamp the incoming video to a known reference level.

3-line mode

The use of certain video sources (for example VCRs) may cause problems in finding the caption lines (Time Base Jitter). However, owing to the 3-line-mode the wandering of the caption line can be noticed without loosing data. In the 3-line mode 3 consecutive lines are sampled and saved in a RAM-buffer via software. Normally the data line lies in the middle, that is, the second recorded line. *In this line only the Run-in key control signal is generated.* If the software cannot detect any caption data there, the line before and the line after are searched for data. If data is to be found there, it is decoded as usual. Also, the software corrects the caption counter ((23EH) so that the caption line lies in the middle of the three lines again. Thus the caption decoder can follow the deviations of the caption line. Naturally this method uses up more RAM space and, in that case, more computing power.

Video-detect mode (see Fig. 2–26)

For Fade and Mute it is necessary for the software to recognize the presence of a video signal. The hardware of the caption decoder can achieve this.

The interrupt service routine for data capture can define a line by writing its number into the caption line register (23EH), and setting the window logic control unit to “video detect mode” by programming bit 1 = ‘1’ of the window logic control register (237H). The processor will receive an interrupt (NMI) at the occurrence of the specified line. The time delayed up to this interrupt is given by evaluating the system counter word in registers 203H and 204H. If no line interrupt occurs, the detection of the next V_{SYNC} that is not derived from the video signal indicates the non-existing video signal. If the time up to the occurrence of the line-interrupt is too short, this also means that no stable video signal is available. *To detect the correct sync-level on the $VIDEO_{IN}$ pin a low-pass filter to ignore the color burst should be applied. The video detection circuitry is designed for an input signal of 1 V_{pp}.*

Software Interface

Address	*Function*	
237H(w)	Window logic control	
	bit 3	H_{sync} select:
		‘0’ = H_{SYNC1}-pin
		‘1 ’= H_{SYNC2}-pin
	bit 2,	‘1’= gated clamp
		‘0’= peak clamp
	bit 1,	‘1’= video det. mode
		(sync detect)
	bit 0,	‘1’= 3-line mode
238H(r/w)	Window logic control	
	bit 7	level Vsync pin (read only)
	bit 6	line counter $\overline{\text{NMI}}$
		‘0’= disabled
		‘1’= enabled
	bit 5	‘1’= CPU-$\overline{\text{SO}}$ input enabled
	bit 4	‘1’= half-dot rounding on
	bit 3	$\overline{\text{OSD active}}$ (read only)
	bit 2	active edge of V_{SYNC}:
		‘0’: rising, ‘1’: falling
	bit 1	active edge of Hsync for acquisition:
		‘0’: rising, ‘1’: falling
	bit 0	active edge of Hsync for OSD:
		‘1’: rising, ‘0’: falling
239H(r/w)	Sync Tip Clamp Start	
	bit 7 to 0 start pos.	
	pos.= start pos.*4/ f_{XTAL}	
23AH(r/w)	Sync Tip Clamp End	
	bit 7...0 end pos.	
	pos.= end pos.*4/ f_{XTAL}	
23BH(r/w)	Run In Key Start	
	bit 7 to 0 start pos.	
	pos.= start pos.*4/ f_{XTAL}	
23CH(r/w)	Run In Key END	
	bit 7 to 0 end pos.	
	pos.= end pos.*4/ f_{XTAL}	
23DH(r/w)	Vertic. Sync Phase (odd/even field detect)	
23EH(r/w)	Caption Line Number	
	$\overline{\text{NMI}}$ is disabled before this register is written for the first time	
23FH(r)	Captured Data	

2.14. OSD

A powerful on-screen display (OSD) is provided on the CCZ 3005K. Many novel approaches were made to save chip area. The most important area saving technique was to eliminate the redundant storage of displayed data. Traditional display devices transfer fixed text data (such as prompts, menus, graphics etc.) from program ROM to the display RAM. The CCZ's OSD, however, is able to display text directly from program ROM. Additionally, the character font table may be located in the program ROM. This offers the ability to size the table exactly as required. Furthermore, specific characters and symbols can be defined. A second character font may be defined in unused portions of RAM which would allow you to create characters "on-the-fly".

The ability of the OSD hardware to access font and display data directly from program ROM was facilitated by incorporating direct memory access (DMA) hardware on chip. A minor slow-down of the processor occurs when the DMA hardware is active. The slow-down of the processor is related to how many characters are on screen and how many color changes occur from character to character.

2.14.1. Summary of OSD Features

- 2 character sizes: 13 H x 8 W or 15 H x 8 W
- soft-scroll
- unlimited numbers of fonts, with any two active at a time
- attributes: flash, italics, transparent, underline
- 8 foreground and 8 background colors
- color palette (8 of 64 colors)
- line-locked display clock
- second color for one text line definable
- very effective for pull-down structures
- half-dot rounding

2.14.2. Fonts

Two different fonts may be defined: one in ROM and one in RAM. The RAM font can be changed by software. This makes it easy to provide foreign language character sets. The basic character set, which is common to all Latin-based languages, for example, can be programmed in the ROM font, containing all characters in the range from ASCII 32 to 127 – i.e., the "printable" ASCII characters. The extensions of the character set that are specific to a language may be contained in the RAM font. These characters can be accessed with offsets to the ROM font or by translating unique ASCII characters with a table.

2.14.3. OSD Window

Start positions of the display are determined and changed (moved) by setting just two register values: Y_Start and X_Start. There is no restriction on text and window size.

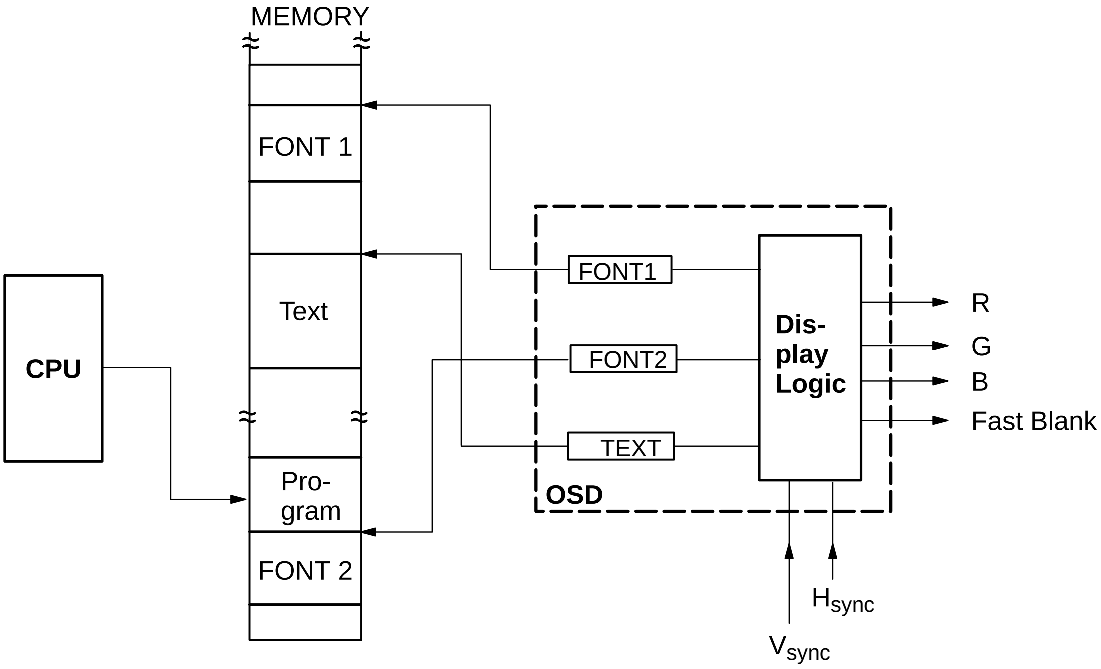


Fig. 2–29: Pointer Model OSD

2.14.4. Colors

2.14.4.1. OSD Attribute ‘COLOR’

‘COLOR’ is a control byte that prefixes the data stream for the OSD. It resides anywhere in the CCZ memory, in ROM or RAM.

Example:

```
   FDB  COLOR    ; define color
   FCC  “String” ; display ‘String’
```

The byte ‘COLOR’ defines the foreground and background color of the characters that follow, until the next ‘COLOR’ attribute is encountered.

OSD attribute ‘COLOR’:

bits 0 to 2:	value 0 to 7 defining foreground color (color 0 to color 7)
bits 3 to 5:	value 0 to 7 defining background color (color 0 to color 7)
bit 6:	‘0’ or ‘1’, if ‘1’: color 0 is replaced by transparent
bit 7:	‘1’: marks color attribute. All other data (characters) have bit 7 = ‘0’.

2.14.4.1.1. Attribute ‘Transparent’

If OSD attribute ‘COLOR’ bit 6 = ‘1’ and background color = ‘000’ but the foreground color is different from ‘000’, the (foreground) character(s) that follow are displayed on a transparent background, i.e., the video source signal is visible instead of the background color. Also the foreground color 0 becomes transparent with OSD attribute ‘COLOR’ bit 6 = ‘1’.

2.14.4.2. Available Colors

The CCZ 3005K has 8 programmable colors. These colors are selectable out of a palette of 64 different values. Each color consists of the 3 components red, green and blue. Each of these components has 1 out of 4 different intensities.

For example, the eight colors could be programmed to be compatible to those of the CCU 3005 C and CCU 3005 D.

Color 0 = black
Color 1 = blue
Color 2 = green
Color 3 = green/blue = cyan
Color 4 = red
Color 5 = red/blue = magenta
Color 6 = red/green = yellow
Color 7 = white

Table 2–6: Color component intensity values

Intensity	MSB	LSB
Off	0	0
1/3	0	1
2/3	1	0
Maximum	1	1

With 4 different intensity values of 3 different colors, 4 · 4 · 4 = 64 different colors are possible.

Table 2–7: Color palette register

R (267H)	G (268H)	B (269H)	Address (Color)	Reset Value
0	0	0	0	Red=Green=Blue=00
0	0	1	1	Red=Green=00; Blue=11
0	1	0	2	Red=00; Green=11; Blue=00
0	1	1	3	Red=00; Green=Blue=11
1	0	0	4	Red=11; Green=Blue=00
1	0	1	5	Red=11; Green=00; Blue=11
1	1	0	6	Red=Green=11; Blue=00
1	1	1	7	Red=Green=Blue=11

2.14.4.3. Color Palette Programming

6 register write accesses are necessary for the 3 palette registers to define the 8 possible colors. To program the 16 bits of each of these 3 registers, every register has to be accessed twice. The first write access programs the least significant 8 bits, and the second write access programs the most significant 8 bits. Each of the 8 programmable colors is defined by three 2-bit intensity values: 2 bits for the red component, 2 bits for green and 2 bits for blue. All least significant bits of these 2 bit values form the first byte to be written (low byte), all MSBs form the second byte (high byte). The bits at location 0 of the low and high byte correspond to color 0, bits at location 1 to color 1, bits at location 2 to color 2, ... and bits at location 7 to color 7.
Example: To get color 0 as light grey, choose 1/3 intensity for each component:

```
1/3 red   = 01
1/3 green = 01
1/3 blue  = 01
```

It is not possible to program a single color on its own, thus all 8 selectable colors have to be programmed together. The other 7 color values have to be defined before ('x' is used instead of any '0' or '1' defining the other colors 1 to 7 in this example).

The programming looks as follows:

The value for 1/3 intensity red = '01'. The least significant bit of this value becomes bit 0 (for color 0) of color palette register red, low byte:

```
LDA   #%xxxxxxx1
STA   color_palette_register_red
```

The most significant bit of intensity red becomes bit 0 (for color 0) of color palette register red, high byte:

```
LDA   #%xxxxxxx0
STA   color_palette_register_red
```

Intensity green = '01', least significant bit becomes bit 0, green, low byte:

```
LDA   #%xxxxxxx1
STA   color_palette_register_green
```

The most significant bit becomes bit 0, green, high byte:

```
LDA   #%xxxxxxx0
STA   color_palette_register_green
```

The same procedure applies to blue:

```
LDA   #%xxxxxxx1  ;the '1' of '01'
STA   color_palette_register_blue
LDA   #%xxxxxxx0  ;the '0' of '01'
STA   color_palette_register_blue
```

A reset changes the color palette to the corresponding colors of the CCU 3005 C and CCU 3005 D (see palette register description in chapter 'Registers').

2.14.4.4. Color Palette Hardware

A common resistor network is used for the generation of the different intensity levels. The output impedance is maximum 2 kOhms. The following diagram shows the working principle. The color palette controls the 12 transistors of the RGB intensity matrix. Only one transistor per output is active at a time.

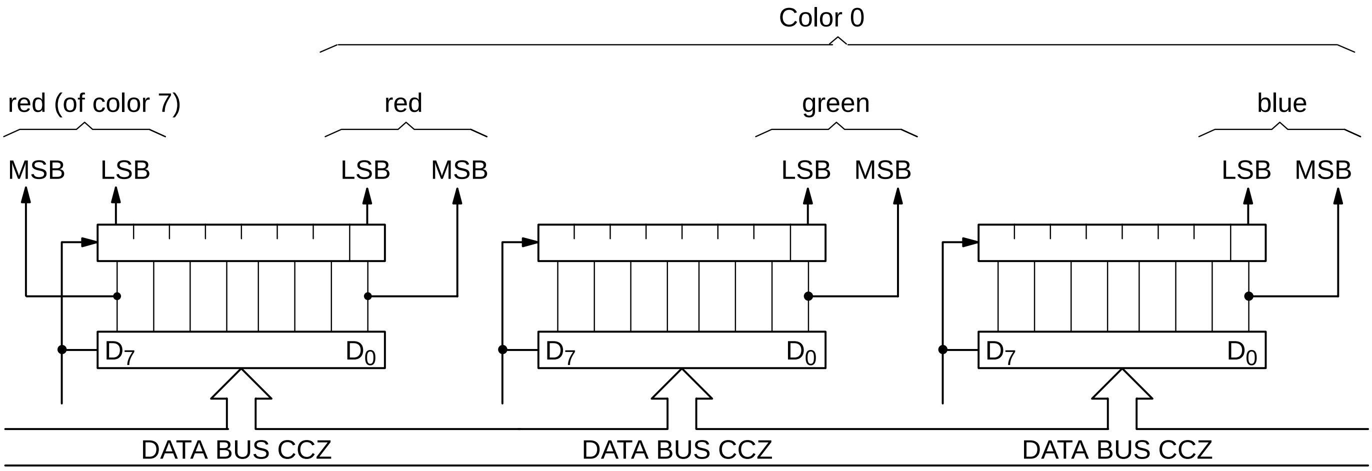


Fig. 2–30: Color map, 3 x 2 bytes

Table 2–8: RGB output levels

Half-Video Level	FB[1]	RGB Level	at V_{SUP} = 4.75V	at V_{SUP} = 5V	at V_{SUP} = 5.25V
‘1’	off	0 %	0V	0V	0V
‘0’	off	0 %	0V	0V	0V
‘0’	on	0 %	0V	0V	0V
‘0’	on	33 %	1,58V	1,66V	1,75V
‘0’	on	66 %	3,16V	3,33V	3,5V
‘0’	on	100 %	4,75V	5V	5,25V

[1] The active level of the Fast Blank output is programmable with bit 7 of the ‘OSD Separate Color Definition Register’ (addr. 266H).

2.14.5. Fast Blank Output

The ‘Fast Blank’ output is active during any OSD activities, i.e.,: the display hardware controlled by the ‘Fast Blank’ must only evaluate the CCZ RGB outputs with ‘Fast Blank’ = active. The polarity of the ‘Fast Blank’ is programmable (‘0’ or ‘1’ active).

2.14.6. Half-Video Output

The OSD generates a ‘Half-Video’ output signal to control external hardware and subdue the video signal. This effects better readability of the OSD, in particular if the video signal’s color and contrast are similar to the OSD signal’s. The ‘Half-Video’ is active instead of ‘color 1’, no matter whether this color 1 is the background or foreground color. Whatever the value of color 1 is defined to be (palette values), the ‘Half-Video’ output becomes ‘1’-level and the RGB outputs become inactive (‘0’-level). The ‘Half-Video’ output can be switched off (set to ‘0’ throughout) by setting bit 0 of the ‘OSD Half-Video Control’ register to ‘1’. After reset, this bit is cleared (‘0’) and the Half-Video Output is active.

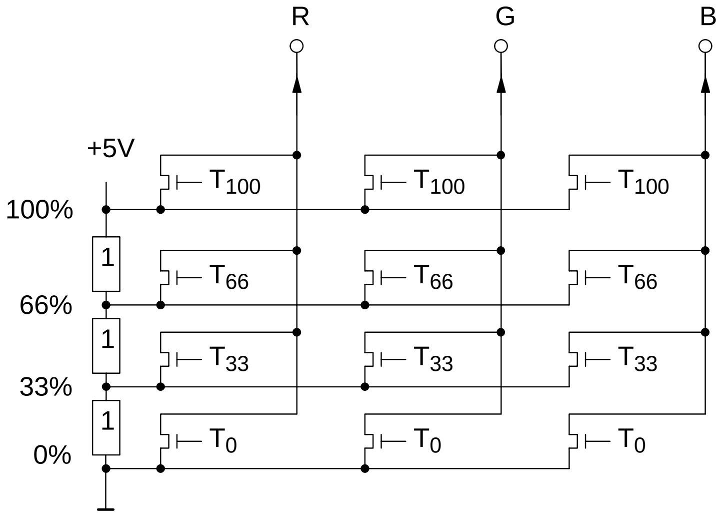


Fig. 2–31: Principle of color palette

2.14.7. Using OSD

The OSD is organized to display character streams (strings), i.e., data streams in the memory are interpreted as OSD attributes or display characters. Via font data the display characters are converted into display pixels. A number of registers offer a convenient method of telling the OSD ‘where’ and ‘what’ shall be displayed. As only 8 bits can be handled at a time, parameters with more than 8 bits in size require multiple writes to the same register, with the MSByte written first. The (unused) high bits of such registers should be set to ‘1’.[1) So, for example, to program 12CH as ‘Last Active scan line’, an “FFH” would first be written into address 261H and then a “2CH”.

To avoid a flickering display, writing to the OSD register should occur synchronized with the vertical synchronization signal.

The active horizontal and vertical pulse width must be at least 6 $T_{OSC,}$ i.e., with a crystal frequency of 6 MHz, it must be greater than or equal to 1 µs.

[1) Disturbing effects caused by internal compare evaluations may appear if unused high bits of values with more than 8 bits are set to ‘0’, and these values are programmed without being synchronized to the horizontal synchronization signal.

Explanation of register functions:

238H(r/w) Window Logic Control Register 1:
bit 7 level Vsync pin (read only)
bit 6 line counter $\overline{\text{NMI}}$
‘0’= disabled
‘1’= enabled
bit 5 ‘1’= CPU–$\overline{\text{SO}}$ input enabled
bit 4 ‘1’= halfdot rounding *) on
bit 3 $\overline{\text{OSD active}}$ **) (read only)
bit 2 active edge of Vsync:
‘0’: rising, ‘1’: falling
bit 1 active edge of Hsync for acquisition:
‘0’: rising, ‘1’: falling
bit 0 active edge of Hsync for OSD:
‘1’: rising, ‘0’: falling

*) Halfdot rounding is not defined for Horizontal Start position (Reg. 0262H) less than 6 and greater than 46.

**) The $\overline{\text{OSD active}}$ bit is set to ‘0’ at the beginning of the ‘First Active Scan Line’. It is set to ‘1’ either after the access to the last scan line of the active display part, defined in ‘Last Active Scan Line’, or after the access to the TEXT-END character.

260H(w) ‘OSD First Active Scan Line’
9 bits:
Specifies the Start Scan Line of the OSD window.

261H(w) ‘OSD Last Active Scan Line’
9 bits:
Determines the last Scan Line of the OSD window to be displayed.
Note that it is possible to set this value to a number that causes the last character lines to be “cut off”. This is a desirable feature when smooth scrolling is in operation.

262H(w) ‘OSD Horizontal Start Position’
6 bits:
Determines horizontal start position of the OSD window in character steps.
(must be greater than 1!).
Don’t use values less than 6 or greater than 46 if ‘Halfdot Rounding’ is enabled (‘Halfdot Rounding’ is enabled if bit 4 in register 0238H = ‘Window Logic Control Register 1’ is set to ‘1’).
The size of one horizontal character step is 8 pixels. (Fine adjust in pixels is possible with ‘OSD Horizontal Start Fine Adjust’, address 26FH).

Remark: The horizontal stop position of the OSD window is determined in the OSD data stream with the attribute ‘CR’ (= 0DH) or ‘END’ (= 0CH).

263H(w) ‘OSD Control Register’, 8 bits:
bit 7 ‘1’ = caption mode
‘0’ = OSD mode
must be set to ‘1’ for caption data.
bit 6 ‘1’ = display active
bit 5 ‘1’ = flash off
‘0’ = flash on
all characters between the attributes ‘FLASH ON’ and ‘FLASH OFF’ are displayed only if this bit is set to ‘0’
bit 4 ‘1’ = 13x8 font
‘0’ = 15x8 font
bit 3..0 first active character Scan Line: determines the start scan line of the first character row of the OSD window. This ability allows the smooth scrolling feature to look correct at the top of the window.

264H(w) ‘OSD Text Start Address Register’, 16 bits: Points to the first character of an OSD data stream. All subsequent characters are encountered until either the OSD attribute ‘END’ or the ‘Vertical Stop Position’ determined in the specific register (address 261H) limits the OSD window.

Remark: The horizontal expansion of the OSD window is determined in character rows which are terminated with ‘CR’ (=0DH).

265H(w) ‘OSD Separate Colored Line’, 9 bits: Start scan line value for a single character line to be displayed with a separate color. Color attributes as part of the OSD data stream (display string) have no effects on the one character high scan lines selected by this register.
The color is defined in the ‘Separate Color Definition Register’.
Thus an entire character line may be highlighted by simply writing its start scan line into this register. This highlighted character line may overlap the boundary of two neighbored ordinary character lines. Thus it can be soft-scrolled through the OSD window.

266H(w) ‘OSD Separate Color Definition Register’ and two more control bits, 8 bits:
bit 7: fast blank output level:
‘1’= active high
‘0’= active low
bit 6: ‘1’= replace black by transparent
‘0’= don’t replace black by transparent
separate color definition:
bits 5 to 3: background color (color no. 0 to 7)
bits 2 to 0: foreground color (color no. 0 to 7)

26AH(w) ‘OSD Half-Video Control Register’:
bit 0: ‘0’= disable half-video (default after RESET), half-video output pin level=‘0’
‘1’=enable half-video

26EH(w) ‘OSD Font 1, Font 2, Start Addresses’, 32 bits: write in the order
MSB font1 (first access)
LSB font 1
MSB font 2
LSB font 2 (last access)
Two pointers to the start of character fonts. Font 1 displays all characters in the range of 0 to 15H, and Font 2 displays all characters in the range between 20H to 7FH. The OSD assumes the address of the first character (= 00H) as font pointer.

26FH(w) ‘OSD Horizontal Start Fine Adjust and Display Modes’:

bit 7	‘1’= horizontal shadow active
bit 6	‘1’= blank display
bits 5..3	not used, set to ‘0’
bits 2..0	horizontal start fine adjust

2.14.8. OSD Attributes

The OSD receives control information from attribute bytes which are part of the OSD data stream. They reside in the CCZ memory (RAM or ROM) together with the text characters to be displayed.

80H to FFH: 'COLOR'
bit 7 '1'= color code (to distinguish this attribute from ASCII or control codes).
bit 6 '1'= color 0 replaced by transparent
bit 5,4,3= background color (color no. 0 to 7)
bit 2,1,0= foreground color (color no. 0 to 7)

01H: 'UNDERLINE_ON'
All subsequent characters are underlined until the OSD attribute 'UNDERLINE_OFF' or 'END' is encountered

02H: 'UNDERLINE_OFF'
See 'UNDERLINE_ON'.

03H: 'FLASH_ON'
The following characters up to the attribute 'FLASH_OFF' or 'END' are displayed only if the 'FLASH' bit in the 'OSD Control Register' is set to '0' (= bit 5, addr. 263H). The flashing occurs only when the flash bit is toggled. This could be done in the interrupt timer function, for example.

04H: 'FLASH_OFF'
See 'FLASH ON'.

05H: 'ITALICS_ON'
All subsequent characters are displayed in italics format until the OSD attribute 'ITALICS_OFF' or 'END'is encountered.

06H: 'ITALICS_OFF'
See 'ITALICS_ON'.

07H: 'TRANSPARENT'
For this character space the underlying video image is shown.

08H: (in standard OSD mode only)
'DOUBLE_UNDERLINE_ON'
like 'UNDERLINE_ON' (01H) but the last two character scan lines are used instead of only the last one.
'UNDERLINE_OFF' (02H) or 'END' turns this mode off again.

09H: (in standard OSD mode only)
'FONT_1 AND_2 '
automatic change of FONT_pointers depending on ASCII-value
(default mode, initialized by RESET and the active V_{SYNC} edge)

0AH: (in standard OSD mode only)
'FONT_1_ONLY'
OSD uses only FONT_1

0BH: (in standard OSD mode only)
'FONT_2_ONLY'
OSD uses only FONT_2

0CH: 'END'
End of the OSD. Your text ***must*** end with this code.

0DH: 'CR'
Carriage return. The following characters are displayed in the next text line.

The OSD insertion must be terminated with CR (0DH) or END (0CH) before the next Hsync!

2.14.9. Font Definition

The OSD has no separate character generator, but the definitions of the characters reside in ROM and/or RAM of the CCZ. Each character is defined by 16 bytes. One byte corresponds to the pixels in the scan line on the screen, the MSB being the first (leftmost) output. The following 15 addresses contain the pixel information for the remaining scan lines of the character. Three bytes for the 13x8 characters and one byte for the 15x8 characters are left unused. For example, the definition of the letter “A” in the 13x8 matrix could look as follows:

bit	7 6 5 4 3 2 1 0	
n	– – – – – – – –	=00H
n+1	– – – **x** – – – –	=10H
n+2	– – **x** – **x** – – –	=28H
n+3	– – **x** – **x** – – –	=28H
n+4	– **x** – – – **x** – –	=44H
n+5	– **x** – – – **x** – –	=44H
n+6	**x** – – – – – **x** –	=82H
n+7	**x x x x x x x** –	=FEH
n+8	**x** – – – – – **x** –	=82H
n+9	**x** – – – – – **x** –	=82H
n+10	**x** – – – – – **x** –	=82H
n+11	– – – – – – – –	=00H
n+12	– – – – – – – –	=00H
n+13	0 0 0 0 0 0 0 0	=00H
n+14	0 0 0 0 0 0 0 0	=00H
n+15	0 0 0 0 0 0 0 0	=00H

OSD insertion must be terminated with ‘CR’ (0DH) or ‘END’ (0CH) before the next Hsync!

The addressing of the pixel pattern to be displayed is given by:

address= font pointer+ASCII*16+scan line

with scan line= 0...12
If the font table does not start with the ASCII character 00H, the font pointer has to be programmed with the corresponding offset. If, for example, the font table starts with the letter “A” (ASCII 65) at address “n”, the the following value results for the font pointer:

font pointer = n–(65*16)

This assumes, of course, that the ASCII representation is used. Font pointer 1 is used to access ASCII characters 00H..1FH, Font pointer 2 is used to access ASCII characters 20H ... 7FH. A single continuous font table results when font pointer 1 and font pointer 2 are set to the same address.

2.14.10. Soft-Scroll

To produce soft-scroll, the software regularly changes the start-scan line of the display (263H, bits 3 to 0). The display of the text line designated by the text pointer starts with this scan line. When the value reaches the last scan line, the text pointer is set to the beginning of the next text line and to the start scan line. Thus the text slowly seems to move upwards. The lower edge is defined by the parameter Y_END. The scroll speed and the direction are determined by how fast the software increments (or decrements) the start scan line register.

To avoid flickering, the change should be effected during the Vsync Interrupt routine.

2.15. Cursor

A cursor can be displayed independently from the OSD output . This cursor is always displayed at the top. It can be of any shape within a field of 24×24 pixels. The logic needs a pointer to a cursor definition bit map. This bit map must be defined somewhere in the CPU address space. The other control registers define the three selected colors, the position of the upper left corner, the fast blank polarity and the half-video or transparent mode. There can be multiple cursor bit maps in memory. Changing the cursor shape just needs a new cursor pointer content.

2.15.1. Cursor Definition

Two masks of 24×24 pixels define the shape. This gives 2 bits per pixel for color definition. If both mask-bits of a cursor pixel are zero, the background (OSD or video) is displayed instead of a cursor pixel. Every other mask-bit combination (01b, 10b and 11b) has a 3-bit color definition register (Col_M1 to Col_M3). The content of the selected register will be used as address for the color palette:

Mask 1 Pixel	Mask 2 Pixel	Displayed Color	Spec. Mode
0	0	background	
1	0	Col_M1	transparent (video)
0	1	Col_M2	1/2 video
1	1	Col_M3	

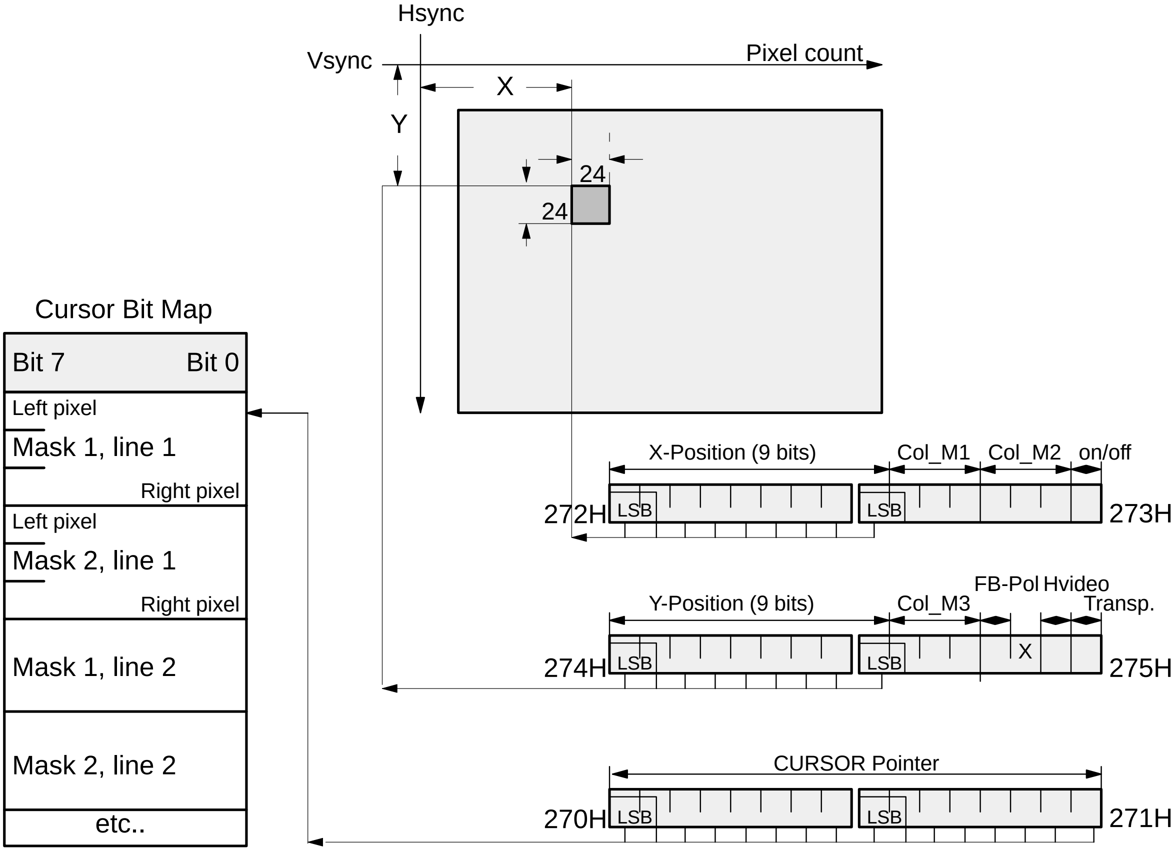


Fig. 2–32: Register model Cursor

The cursor logic reads the bitmap using DMA cycles. There is a 16-bit pointer to a block of 3*2*24 = 144 bytes. For every cursor scanline the logic needs 6 bytes: 3 bytes(= 24 pixels) for mask1, and 3 bytes for mask2. This explains the cursor definition format:

Cursor pointer →	line1, mask1, byte1	address N
	line1, mask1, byte2	N+1
	line1, mask1, byte3	N+2
	line1, mask2, byte1	N+3
	line1, mask2, byte2	N+4
	line1, mask2, byte3	N+5
	line2, mask1, byte1	N+6
	line2, mask1, byte2	N+7
	line2, mask1, byte3	N+8
	line2, mask2, byte1	N+9
	line2, mask2, byte2	N+10
	line2, mask2, byte3	N+11
	etc..	

Even if the OSD has the character rounding active, the cursor logic displays the same 24*24 pixels on both fields. The visible cursor field is therefore (on interlaced displays) 24 pixels*48 scan lines wide.

2.15.2. Cursor Position

The cursor can be positioned with pixel resolution. Two 9-bit registers define the X and Y position of the upper left corner of the cursor field. The logic uses the same display clock, H-sync and V-sync signals as the OSD. This guarantees a perfect match between OSD and Cursor positions. The OSD has a character-based X and a scanline-based Y position scheme, the cursor needs positions in pixel resolution. Note: the X-fine Register in the OSD acts as an offset of the OSD_Hsync. Changing this register will move both, the OSD and the cursor. It is not necessary to include the x-fine position into the cursor position calculation.

2.15.3. Moving and Changing the Cursor

The cursor logic has 6 I/O registers. Changing only the cursor position requires 2 to 4 write operations, changing the shape and the position needs 6 writes. The content of the registers from the first write to the last write is intermediate and can therefore produce jumping effects, wrong colors or wrong cursor shapes. To avoid these effects, disable the cursor with the first write and synchronize your cursor control software with Vsync_OSD. Enable the cursor again as last write to the corresponding control register. Doing all the changes after Vsync gives enough time to complete the modifications before the first cursor scanline is reached.

2.15.4. Cursor Control Bits

The color ‘Col_M1’ can be replaced by transparent, the color ‘Col_M2’ by 1/2 video, if the corresponding controlbit is set. Instead of the cursor pixel the video pixel will be displayed (with reduced intensity for ‘1/2 video’), even if the cursor is over an OSD area. This is different from the ‘background’ color (both mask bits=0): ‘background’ displays the background of the cursor (OSD or video), ‘transparent’ always uses the video as color.

The FB_polarity bit selects the active state of the FB signal during the display of the cursor field. This flag must be set to the same level as the FB_select bit in the OSD. The on/off flag allows the software to enable – or disable the cursor.

2.15.5. Cursor DMA

The cursor and the OSD access the CPU memory via DMA. The DMA of the cursor logic is done right after the OSD-H-sync of a cursor-scanline (this is a scanline that shows cursor pixels, 24 lines per field). Six subsequent DMA cycles transfer the two mask patterns for one cursor scanline into the display buffer of the cursor logic. Collisions with the OSD-DMA are avoided since it is not allowed to select OSD positions less than 1 (without rounding) or 6 (with rounding): the DMA cycles are completed before the OSD can become active. (Note: OSD-X-positions count in character steps, an X-position increment of one is equivalent to 8 bus cycles).

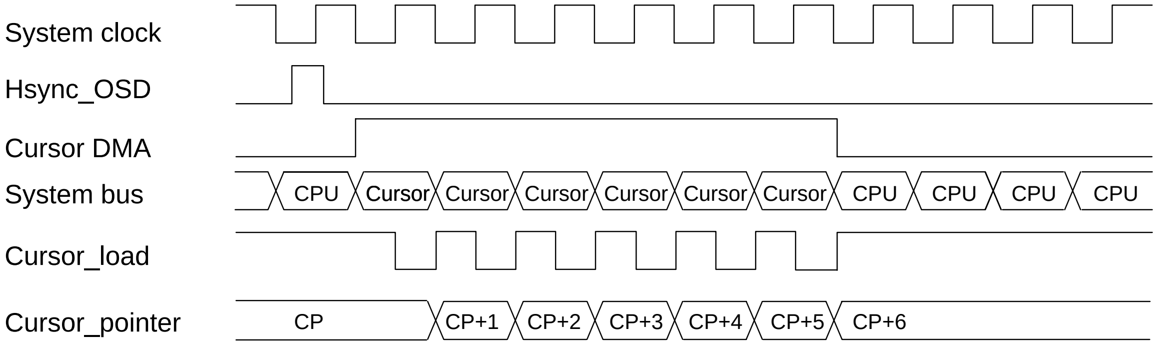


Fig. 2–33: Cursor DMA timing

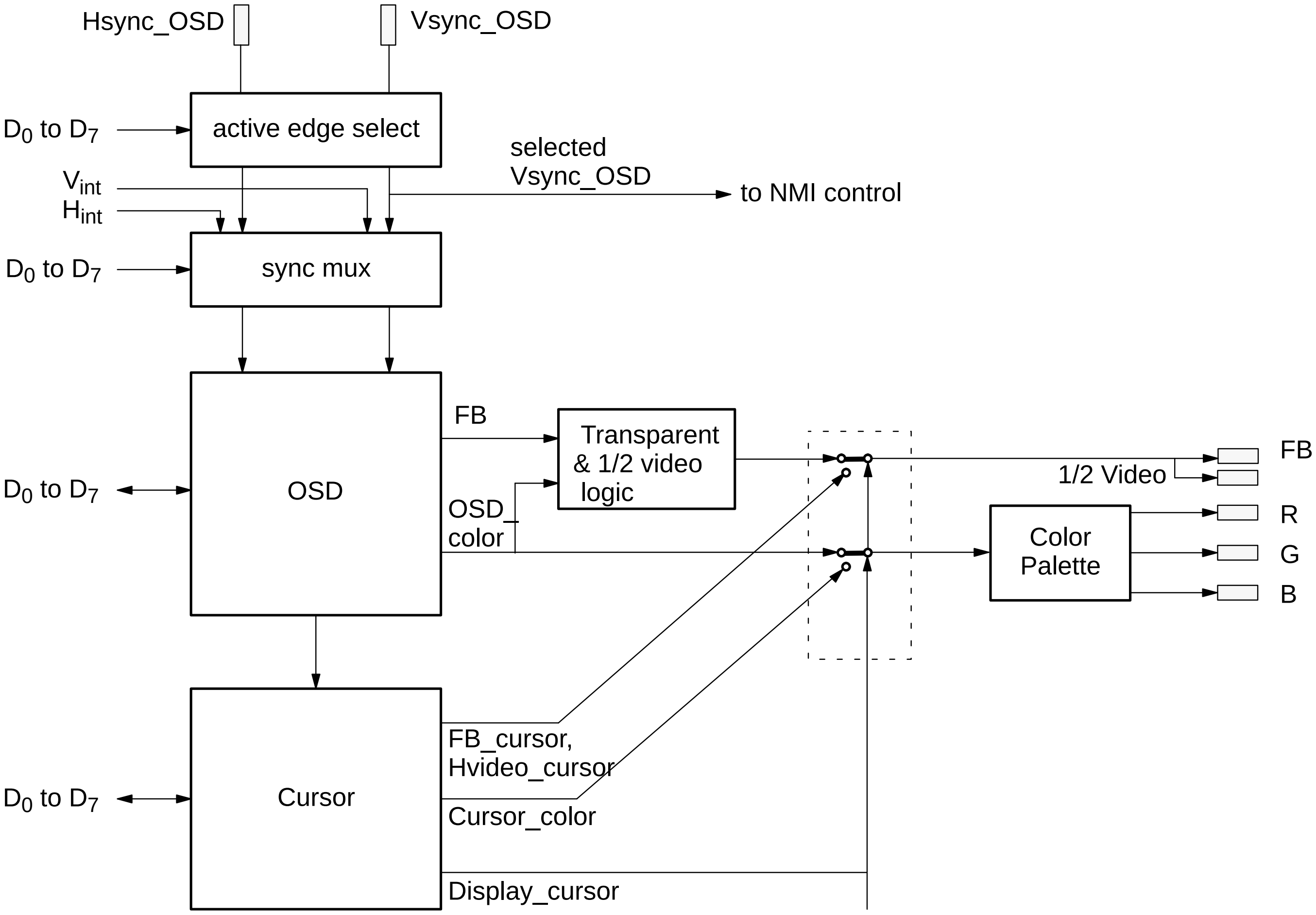


Fig. 2–34: OSD and Cursor block diagram

2.16. H & V Sync Generator

The OSD and cursor logic of CCZ 3005K needs horizontal and vertical synchronization inputs to generate the internal timing signals. If these sync inputs are not stable (e.g. during channel search) the OSD will not have a stable position on the TV screen.

The H & V sync generator delivers stable horizontal and vertical sync pulses which can be used to synchronize the OSD. Via a programmable sync multiplexer the OSD can be connected either to the H_{SYNC} and V_{SYNC} input pins or the output signals of the H & V sync generator (see Fig. 2–35). **The switching of the sync multiplexer should only occur when the OSD is inactive!** After reset the OSD is connected to the H_{SYNC} and V_{SYNC} input pins.

The H & V sync generator can be used either in free running or in tracking mode. In free running mode the H & V sync generator simply counts with $\mathbf{f_{system}}$ and generates syncs at fixed positions (H = 384, V = 312/262). In tracking mode the sync generator tries to follow the H_{SYNC} and V_{SYNC} input signals, but only within programmable limits. Both horizontal and vertical tracking can be programmed independently. If tracking is enabled, a 5–bit speed value can be set which defines how fast the internal counters will be adapted (for example: H = 384±10, V = 312±5).

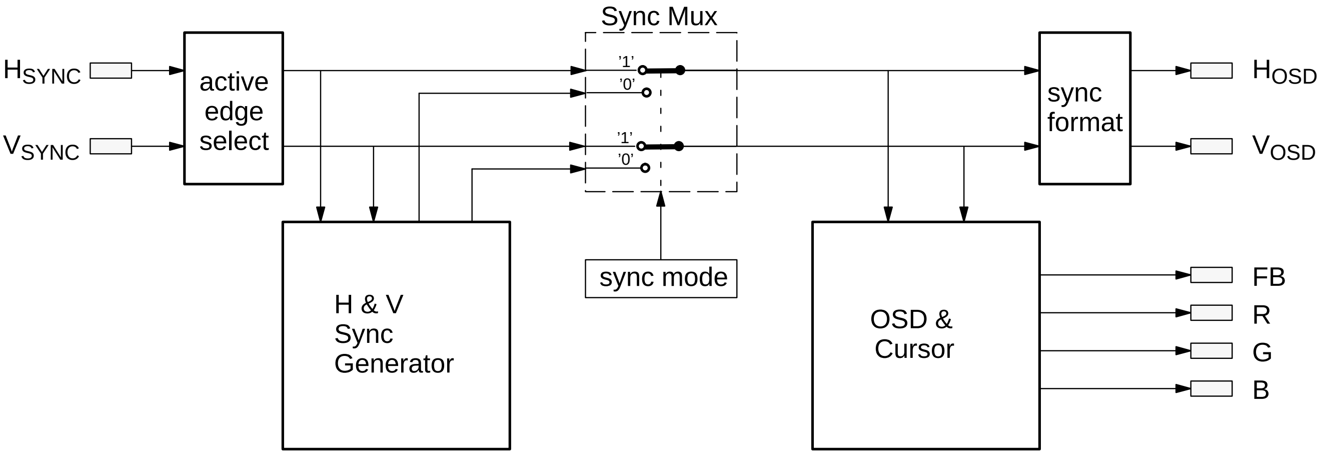


Fig. 2–35: OSD synchronization

The output signals of the H & V sync generator can be programmed in various ways to adapt to different applications. Both outputs can be inverted independently and can be combined into a composite sync signal. The vertical field length can be set to PAL or NTSC mode.

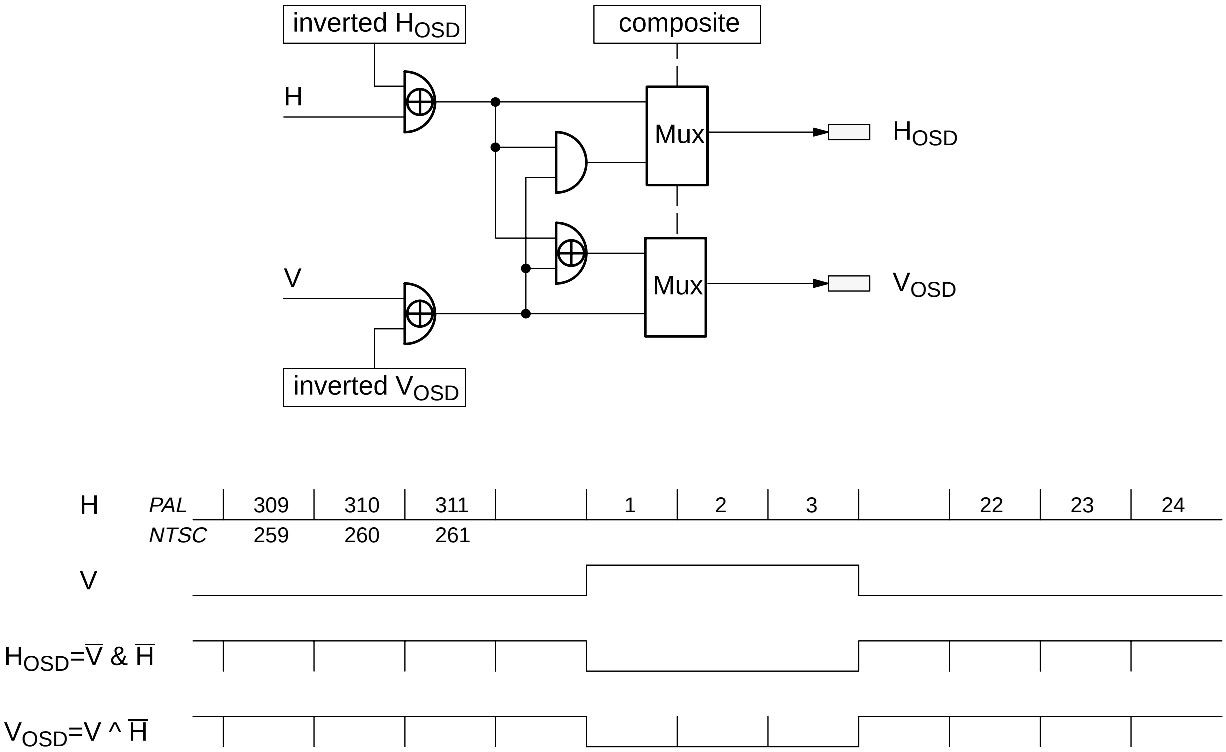


Fig. 2–36: Programming sync format of H_{OSD} and V_{OSD}

2.17. Infrared Input

Programmed to work in special input mode, P35 (bit 5 of port 3) may be used to detect infrared signals.

The hardware of the infrared input is designed to support software decoding of different infrared signals. In most infrared telegrams, data are coded as pulse sequences or different logical levels that follow start edges or pulses. A logical ‘0’ differs from a ‘1’ in the availability of a defined logical level (‘0’ or ‘1’) during specific time slots after this start edge or pulse, in defined delays between the start edge (or pulse) and a followed data pulse or a logical level during some time after the start edge.

The hardware of the CCZ 3005K offers the possibility to program two time values that define the moments when the infrared signal is scanned. Two values of 0 to 14 as nibbles in the 8-bit ‘IR Sample Times’ register determine two delays starting at the active infrared signal edge. The value 15 for ‘IR sample times’ is not allowed. With bit 5 in the ‘IR Control and Status’ register, the step size of the sample time value is programmable: with bit 5 set to ‘0’, the step size is 85 µs and values from 0 to 14 x 85 µs = 1.19 ms are determined. With bit 5 = ‘1’, the step size is 170 µs and the values are from 0 to 14 x 170 µs = 2.38 ms. The absolute delay error is +1 LSB, for example. With step size = 85 µs and sample time value = 2, the delay is in the range between 170 µs and 255 µs.

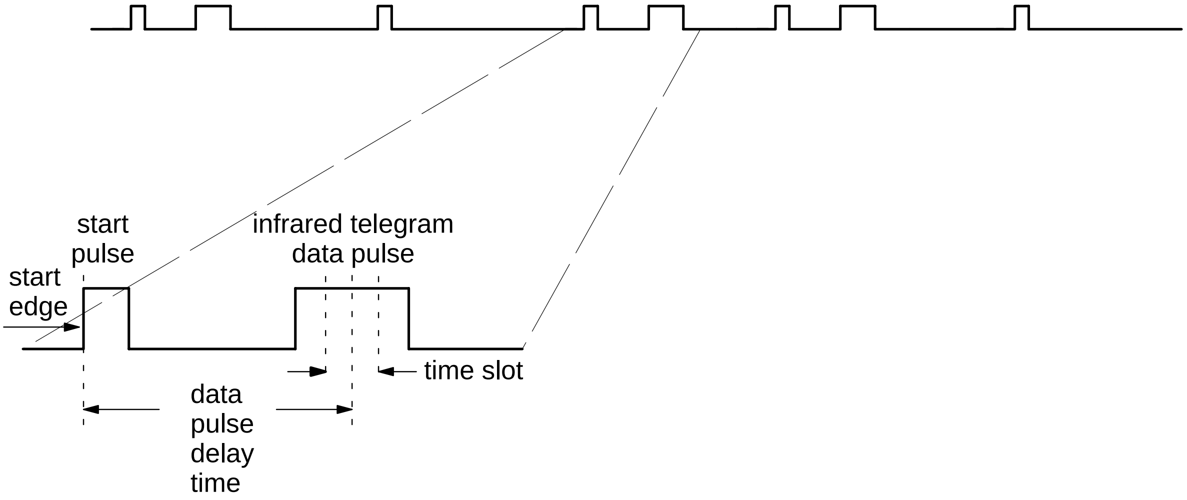


Fig. 2–37: Part of an infrared telegram

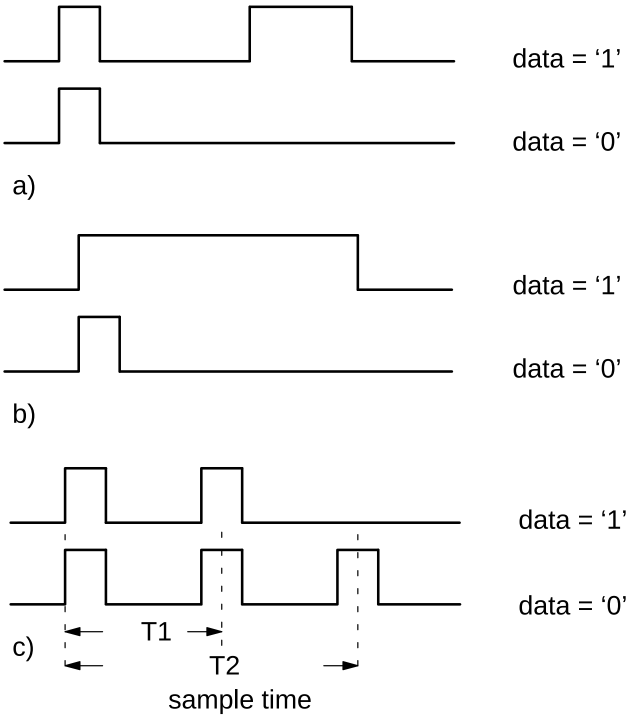


Fig. 2–38: Examples of differently coded infrared signals

In the ‘IR Control and Status’ register, it is programmable whether a rising or falling edge has to be evaluated as active infrared signal start edge and whether infrared detection has to generate interrupts. Either the active infrared signal start edge or the detection of the higher of both programmed sampling times (‘both samples taken’) may generate an interrupt. The result of scanning is delivered in the ‘IR Control and Status’ register.

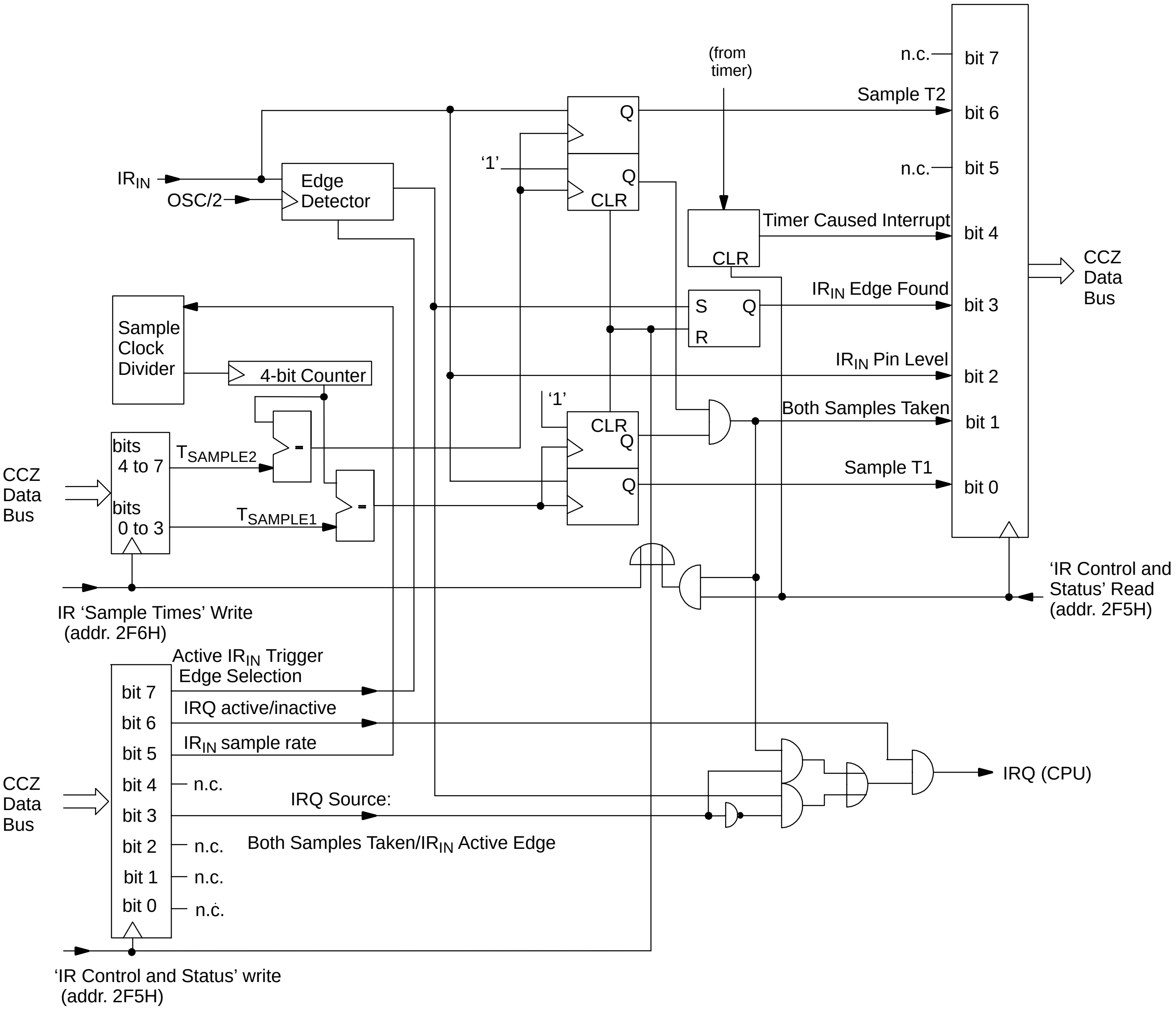


Fig. 2–39: Block diagram of infrared input

2.17.1. Infrared Detection Status

Infrared Status Register as Part of the ‘IR Control and Status Register’ (addr. 2F5H):

Bit 6 = IR_{IN} pin level at moment T2
Bit 4 = Interrupt source:
If ‘1’: timer caused interrupt
(2.048 ms if system clock = 12 MHz)
Bit 3 = If ‘1’: IR_{IN} active start edge found
Bit 2 = IR_{IN} pin level
(direct, as if read by standard
input port)
Bit 1 = If ‘1’: both samples taken
Bit 0 = IR_{IN} pin level at moment T1

Bit 4 ‘interrupt source’ is set to ‘0’ after every read access to the ‘IR Control and Status’ register, while bits 0, 1, 3 and 6 become ‘0’ after reading the status with a ‘1’ in bit 1, indicating that both samples have been taken, i.e.,: bits 0 and 6 are valid. To trigger the infrared signal start edge only, and not the data edges as well, the sample counters are not retriggerable. They start with the first occurrence of the edge they are programmed for (rising or falling) and become sensitive for the next start after the higher of both sampling times has passed (‘both samples taken’), so, in the worst case, after 2.55 ms.

2.17.2. Infrared Detection Control

By writing to the ‘IR Control and Status’ register, the active start edge of the IR_{IN} signal can be determined:

bit 7 = ‘0’: use first detected falling edge,
bit 7 = ‘1’: use first detected rising edge

of the IR_{IN} signal to start the two sample timers.

If and what can generate an interrupt is also selectable:

Bit 6 = ‘0’: disable IRQ
Bit 6 = ‘1’: enable IRQ .

This IRQ can be caused by the chosen active edge (selected with bit 7) or if both samples are taken, (in parallel to bit 1 of the status register):

bit 3 = ‘0’: use active IR_{IN} edge,
bit 3 = ‘1’: generate IRQ when both samples are taken
(same signal as in status register).

Bit 5 of the control part of the ‘IR Control and Status’ register determines the step size for the programmable sample rate:

bit 5 = ‘0’: step size = 85 µs,
bit 5 = ‘1’: step size = 170 µs.

Programming the ‘IR Control and Status’ register may already cause an IRQ after having been enabled by the processors’ command ‘CLI’ before. Also the ‘CLI’ that follows the write to the register may cause the IRQ if it was not cleared by reading this register before (refer to chapter 2.19.).

2.17.3. Sample Times

With two nibbles in the ‘IR Sample Times’ register (addr. 2F6H), two moments for sampling the IR_{IN} signal are programmable. The values between 0 and 14 have to be multiplied with either 85 µs or 170 µs, depending on how bit 5 of the ‘IR Control and Status’ register is programmed. Thus values between 0 and 14 x 170 µs = 2.38 ms are programmable.

2.18. Timer

The IRQ input is accessed by a free-running timer with 2.048 ms (f_{system} = 6 MHz). As soon as an IRQ has been generated, each further IRQ depends on the reading of the infrared register, which has to be done first and which also provides information on the IRQ-source (IR-input is also an IRQ-source) (see fig. 2–39).

The readable system counter is derived from the 16-bit prescaler. Although this counter cannot be preset, it can be read. Its value is incremented by 1 with every $\Phi 2$ clock. Overrun occurs from FFFFH to 0000H. To get a definite value, the MSByte is copied into a register that can later be read out (204H) while the LSByte (203H) is read; i.e., the MSByte is latched by reading the LSByte.

Address	*Function*
203H(r)	LSByte timer, save MSByte
204H(r)	MSByte (saved value)

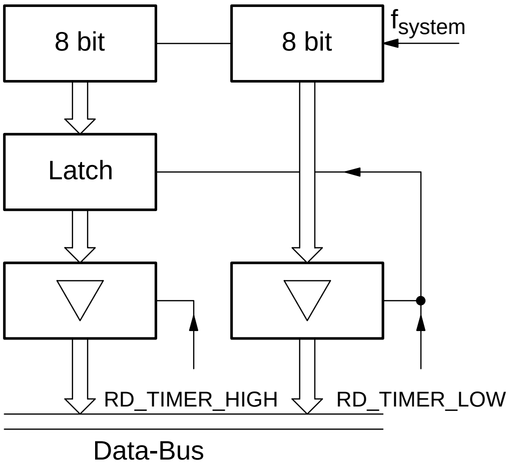


Fig. 2–40: Readable Timer

2.19. Interrupt System

The CPU 65C02 contains two interrupts.

Interrupt	*Sources*
– IRQ	IR-Interface, Timer
– NMI	VSYNC, Caption line

The IRQ is mask-programmable via software. The infrared register detects which of the two sources has triggered the interrupt. The IR interface can also separately be switched off as INT source. As soon as an IRQ is generated, it is disabled until the ‘IR Control and Status’ register is read. If the ‘IR Control and Status’ register is read while an enabled IRQ is pending, the IRQ can be ignored. The software is responsible for starting the IRQ function if an IRQ is detected.

The NMIs also become disabled with occurrence. The sources are distinguished by testing the VSYNC-input, bit 7 of the ‘Window Logic Control Register 1’. To get a further NMI, a WRITE with any value to the ‘NMI Return’ register (addr. 236H) has to be executed immediately before the “RTI” of the NMI service routine. As WRITEs to the ‘NMI Return’ register are registered, the first WRITE should not occur before the first occurrence of an NMI. *Therefore it is not advisable to execute a WRITE to* the *‘NMI Return’ during initialization, because in that case, the first NMI does not block itself and the second NMI may be nested!* A WRITE (of any data) to the ‘NMI Return’ register enables a succeeding NMI with the next processor SYNC signal.

The NMI for both the V_{SYNC} and caption line is disabled until a first WRITE to the ‘Closed Caption Line Number’ register (address 23EH) occurs.

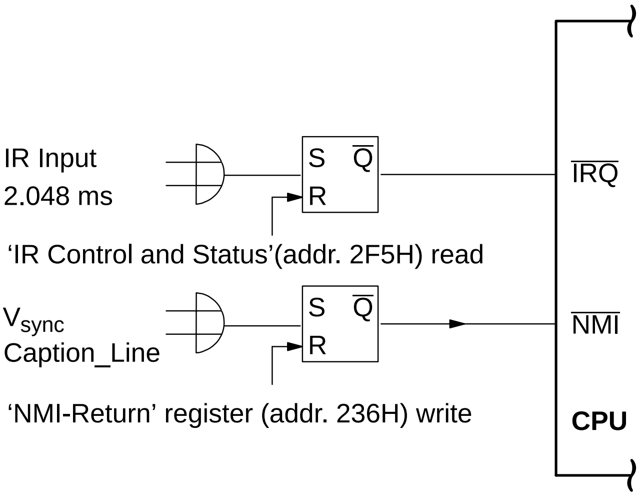


Fig. 2–41: INT system CCZ 3005K

3. Specifications

3.1. Outline Dimensions

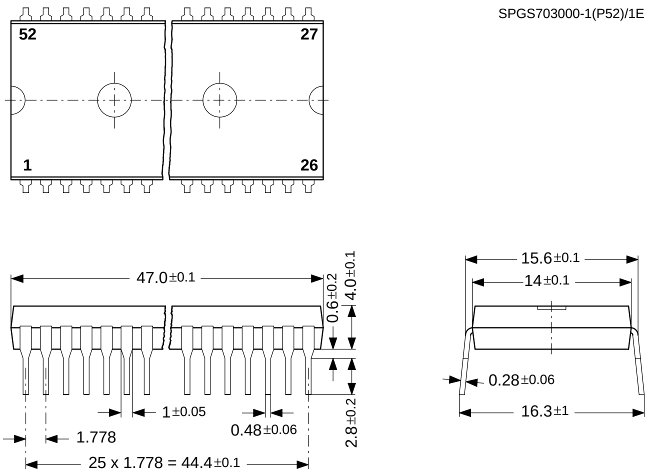


Fig. 3–1:
52-Pin Plastic Shrink Dual-In-Line Package
(PSDIP52)
Weight approximately 5.5 g
Dimensions in mm

3.2. Pin Connections and Short Descriptions

X = obligatory; connect as described in circuit diagram

Pin No. PSDIP 52-pin	Pin Name	Type	Connection (if not used)	Short Description
1	P34	IN/OUT	X	Port 3, Bit 4
2	P35 /IR_{IN}	IN/OUT	X or output signal of ext. infrared receiver	Port 3, Bit 5 or infrared signal input
3	P36	IN/OUT	X	Port 3, Bit 6
4	P37/PWM6	IN/OUT	X	Port 3, Bit 7 or pulse width modulator (D/A converter) output no. 6
5	P00	IN/OUT	X	Port 0, Bit 0
6	P01	IN/OUT	X	Port 0, Bit 1
7	P02	IN/OUT	X	Port 0, Bit 2
8	P03	IN/OUT	X	Port 0, Bit 3
9	P04/V_{OSD}	IN/OUT	X	Port 0, Bit 4 or Vertical synchronization output of internal H&V sync generator
10	P05/H_{OSD}	IN/OUT	X	Port 0, Bit 5 or Horizontal synchronization output of internal H&V sync generator
11	P06	IN/OUT	X	Port 0, Bit 6

Pin Connections and Short Descriptions, continued

Pin No. **PSDIP 52-pin**	**Pin Name**	**Type**	**Connection** (if not used)	**Short Description**
12	P07	IN/OUT	X	Port 0, Bit 7
13	VIDEO$_{IN}$	IN	external video source signal	Input for external video signal (for Closed Caption data decoding)
14	SLICER CAPACITANCE	IN/OUT	slicer capacitor	Connection for slicer capacitor
15	PORQTEST	IN	GND	Test pin (usable by manufacturer only)
16	ADC0	IN	external analog signal	Analog converter input no. 0
17	ADC1	IN		Analog converter input no. 1
18	ADC2	IN		Analog converter input no. 2
19	ADC3	IN		Analog converter input no. 3
20	ADC4	IN		Analog converter input no. 4
21	GNDA	SUPPLY	GND	Ground connection (same as digital ground)
22	V$_{SUPA}$	SUPPLY	+5V analog	Analog power supply
23	HALF-VIDEO	OUT	Half-Video control input of external RGB stage	Half-Video control output of internal OSD circuit
24	B$_{OUT}$	OUT	blue input of external RGB stage	Blue color control output of internal OSD circuit
25	G$_{OUT}$	OUT	green input of external RGB stage	Green color control output of internal OSD circuit
26	R$_{OUT}$	OUT	red input of external RGB stage	Red color control output of internal OSD circuit
27	FAST BLANK	OUT	Fast Blank input of external RGB stage	Fast Blank control output of internal OSD circuit
28	H$_{SYNC1}$	IN	H$_{SYNC}$ output of external OSD stage	Horizontal synchronization input of internal OSD circuit and Closed Caption Detection circuit
29	V$_{SYNC}$	IN	V$_{SYNC}$ output of external OSD stage	Vertical synchronization input of internal OSD circuit

Pin Connections and Short Descriptions, continued

Pin No. PSDIP 52-pin	Pin Name	Type	Connection (if not used)	Short Description
30	P10 / IM$_2$-DAT / I^{2}C$_2$-SDA	IN/OUT	X or I^{2}C-data line of external device(s)	Port 1, Bit 0 or I^{2}C-data
31	P11 / IM$_2$-CLK / I^{2}C$_2$-SCL	IN/OUT	X or IM-ident line of external device(s)	Port 1, Bit 1 or I^{2}C-clock
32	P12 / IM-ID	IN/OUT		Port 1, Bit 2 or IM-ident
33	P13 / IM$_1$-DAT / I^{2}C$_1$-SDA	IN/OUT	X or IM-data resp. I^{2}C-data line of external device(s)	Port 1, Bit 3 or IM-data resp. I^{2}C$^-$data
34	P14 / IM$_1$-CLK / I^{2}C$_1$-SCL	IN/OUT	X or IM-clock resp. I^{2}C-clock line of external device(s)	Port 1, Bit 4 or IM-clock resp. I^{2}C$^-$clock
35	$\overline{\text{RESET}}$	IN/OUT	external reset signal	CCU reset signal
36	XTAL1	IN	CCU clock crystal	Crystal connector
37	XTAL2	OUT		Crystal connector
38	GND	SUPPLY	GND	Ground connection
39	V$_{SUP}$	SUPPLY	+5 V (digital)	Power supply (digital)
40	TEST	IN	GND	Test pin (usable by manufacturer only)
41	P20 / PWM0	IN/OUT	X or low pass filter and analog input of external device	Port 2, Bit 0 or pulse width modulator (D/A converter) output no. 0
42	P21 / PWM1	IN/OUT		Port 2, Bit 1 or pulse width modulator (D/A converter) output no. 1
43	P22 / PWM2	IN/OUT		Port 2, Bit 2 or pulse width modulator (D/A converter) output no. 2
44	P23 / PWM3	IN/OUT		Port 2, Bit 3 or pulse width modulator (D/A converter) output no. 3
45	P24 / PWM4	IN/OUT		Port 2, Bit 4 or pulse width modulator (D/A converter) output no. 4
46	P25 / PWM5	IN/OUT		Port 2, Bit 5 or pulse width modulator (D/A converter) output no. 5

Pin Connections and Short Descriptions, continued

Pin No. **PSDIP 52-pin**	**Pin Name**	**Type**	**Connection** (if not used)	**Short Description**
47	P26	IN/OUT	X	Port 2, Bit 6
48	P27 / H_{SYNC1}	IN/OUT	X or H_{SYNC} output of external OSD stage	Port 2, Bit 7 or horizontal synchronization input of internal OSD circuit and Closed Caption Detection circuit
49	P30	IN/OUT	X	Port 3, Bit 0
50	P31	IN/OUT	X	Port 3, Bit 1
51	P32	IN/OUT	X	Port 3, Bit 2
52	P33	IN/OUT	X	Port 3, Bit 3

3.3. Pin Descriptions

Pin numbers refer to the 52-pin PSDIP package. The functions of some port pins can be changed to either ‘Standard Mode’ or ‘Special Mode’ by setting the specific bits in their mode registers.

Pin 1: P34:
Bit 4 of port 3,
connection depends on application

Pin 2: P35 or IR_{IN}:
in *Standard Mode*:
Bit 5 of port 3,
connection depends on application
in *Special Mode*:
Infrared signal input, to connect with the output of the infrared receiver

Pin 3: P36:
Bit 6 of port 3,
connection depends on application

Pin 4: P37 or PWM6:
in *Standard Mode*:
Bit 7 of port 3,
connection depends on application
in *Special Mode*:
PWM (D/A converter) output no. 6

Pin 5: P00:
Bit 0 of port 0,
connection depends on application

Pin 6: P01:
Bit 1 of port 0,
connection depends on application

Pin 7: P02:
Bit 2 of port 0,
connection depends on application

Pin 8: P03:
Bit 3 of port 0,
connection depends on application

Pin 9: P04 or V_{OSD}:
in *Standard Mode*:
Bit 4 of port 0,
connection depends on application
in *Special Mode*:
vertical synchronization output of internal H&V sync generator

Pin 10: P05 or H_{OSD}:
in *Standard Mode*:
Bit 5 of port 0,
connection depends on application
in *Special Mode*:
horizontal synchronization output of internal H&V sync generator

Pin 11: P06
Bit 6 of port 0
connection depends on application

Pin 12: P07
Bit 7 of port 0
connection depends on application

Pin 13: $Video_{IN}$:
Video signal input to connect the closed caption data signal

Pin 14: Slicer Capacitance:
Slicer Capacitance connector, with the other side of the capacitor to GND

Pin 15: GND (PORQTEST):
has to be connected to GND
(usable by manufacturer only)

Pin 16: ADC0:
Analog to digital converter input no. 0

Pin 17: ADC1:
Analog to digital converter input no. 1

Pin 18: ADC2:
Analog to digital converter input no. 2

Pin 19: ADC3:
Analog to digital converter input no. 3

Pin 20: ADC4:
Analog to digital converter input no. 4

Pin 21: GNDA:
Analog ground (GND) input

Pin 22: V_{SUPA}:
Analog voltage supply (+5V) input

Pin 23: Half-Video:
Half-video output signal, to control the OSD output stage working in half-video mode

Pin 24: $B_{OUT:}$
Blue intensity output, to control the OSD output stage

Pin 25: G_{OUT}
Green intensity output, to control the OSD output stage

Pin 26: R_{OUT}:
Red intensity output, to control the OSD output stage

Pin 27: Fast Blank:
Fast Blank output, to enable the OSD output stage using the CCZ’s R, G, B outputs

Pin 28: H_{SYNC1}:
One of two available horizontal synchronization signal inputs to connect with the horizontal synchronization output signal of the external OSD source

Pin 29: V_{SYNC}:
Vertical synchronization input signal to connect with the horizontal synchronization output signal of the external OSD source.

Pin 30: P10 or IM_2-DAT resp. I^2C_2-SDA:
in *Standard Mode:*
Bit 0 of port 1, connection depends on application
in *Special Mode:*
IM-bus data or I^2C-bus data, to connect with the same signal(s) of external device(s). It depends on the register that was written to (IM-bus control or I^2C-bus control) whether the terminal is used as IM-bus or as I^2C-bus data line.

Pin 31: P11 or IM_2-CLK resp. I^2C_2-SCL:
in *Standard Mode:*
Bit 1 of port 1, connection depends on application
in *Special Mode:*
IM-bus clock or I^2C-bus clock, to connect with the same signal(s) of external device(s). It depends on the register that was written to (IM-bus control or I^2C-bus control register) whether the terminal is used as IM-bus or as I^2C-bus clock line.

Pin 32: P12 or IM-ID:
in *Standard Mode:*
Bit 2 of port 1, connection depends on application
in *Special Mode:*
IM-bus ident, to connect with the same signal(s) of external device(s).

Pin 33: P13 or IM_1-DAT resp. I^2C_1-SDA:
in *Standard Mode:*
Bit 3 of port 1, connection depends on application
in *Special Mode*
IM-bus data or I^2C-bus data, to connect with the same signal(s) of external device(s). It depends on the register that was written to (IM-bus control or I^2C-bus control) whether the terminal is used as IM-bus or as I^2C-bus data line.

Pin 34: P14 or IM_1-CLK resp. I^2C_1-SCL:
in *Standard Mode:*
Bit 4 of port 1, connection depends on application
in *Special Mode:*
IM-bus clock or I^2C-bus clock, to connect with the same signal(s) of external device(s). It depends on the register that was written to (IM-bus control or I^2C-bus control register) whether the terminal is used as IM-bus or as I^2C-bus clock line.

Pin 35: $\overline{\text{RESET}}$:
CCZ reset pin to connect with a signal to reset the CCZ.

Pin 36: XTAL1:
First crystal connector.

Pin 37: XTAL2:
Second crystal connector

Pin 38: GND:
Digital ground (GND) input

Pin 39: VSUP:
Digital Voltage supply (+5V) input

Pin 40: TEST:
Test input, leave vacant or connect with GND

Pin 41: P20 or PWM0:
in *Standard Mode:*
Bit 0 of port 2,
connection depends on application
in *Special Mode:*
PWM (D/A converter) output no. 0

Pin 42: P21 or PWM1:
in *Standard Mode:*
Bit 1 of port 2,
connection depends on application
in *Special Mode:*
PWM (D/A converter) output no. 1

Pin 43: P22 or PWM2:
in *Standard Mode:*
Bit 2 of port 2,
connection depends on application
in *Special Mode:*
PWM (D/A converter) output no. 2

Pin 44: P23 or PWM3:
in *Standard Mode:*
Bit 3 of port 2,
connection depends on application
in *Special Mode:*
PWM (D/A converter) output no. 3

Pin 45: P24 or PWM4:
in *Standard Mode:*
Bit 4 of port 2,
connection depends on application
in *Special Mode:*
PWM (D/A converter) output no. 4

Pin 46: P25 or PWM5:
in *Standard Mode:*
Bit 5 of port 2,
connection depends on application
in *Special Mode:*
PWM (D/A converter) output no. 5

Pin 47: P26:
Bit 6 of port 2,
connection depends on application

Pin 48: P27 or H_{SYNC2}:
in *Standard Mode:*
Bit 7 of port 2,
connection depends on application
in *Special Mode:*
One of two available horizontal synchronization signal inputs to connect with the horizontal synchronization output signal of the external OSD source

Pin 49: P30:
Bit 6 of port 3,
connection depends on application

Pin 50: P31:
Bit 1 of port 3,
connection depends on application

Pin 51: P32:
Bit 2 of port 3,
connection depends on application

Pin 52: P33:
Bit 3 of port 3,
connection depends on application

3.4. Pin Configuration

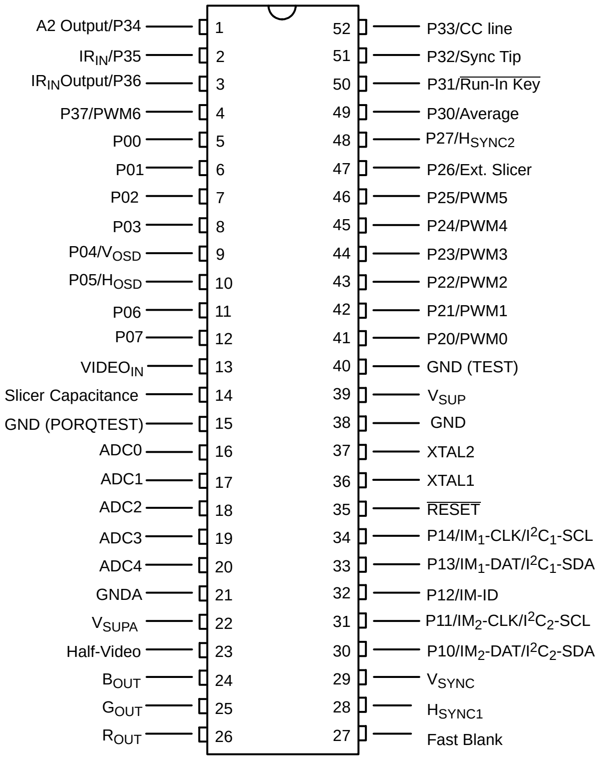


Fig. 3–2: 52-pin PSDIP package, top view

3.5. Pin Circuits

Pin numbers refer to 52-pin PSDIP package.

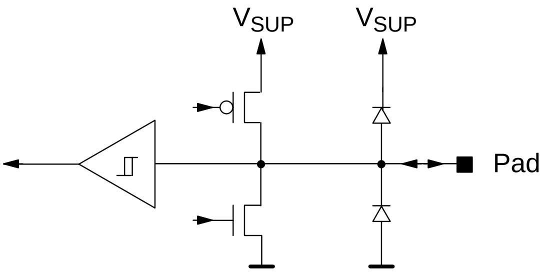


Fig. 3–3: Push-Pull I/O

P30 to P37:	Pins 1 to 4 and 49 to 52
P20 to P27:	Pins 41 to 48
P12:	Pin 32
P00 to P07:	Pins 5 to 12.

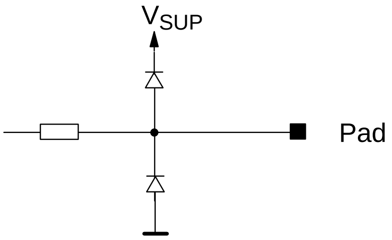


Fig. 3–7: Analog input

XTAL1:	Pin 36
ADC0 to ADC4:	Pins 16 to 20

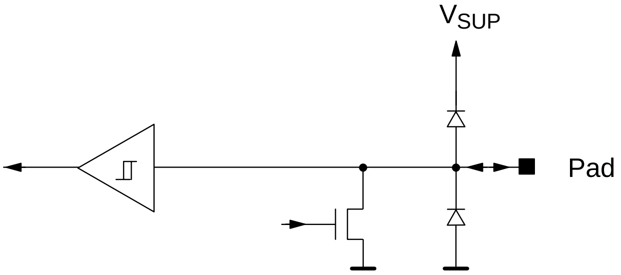


Fig. 3–4: Open drain I/O
P10, P11, P13, P14: Pins 30, 31, 33, 34

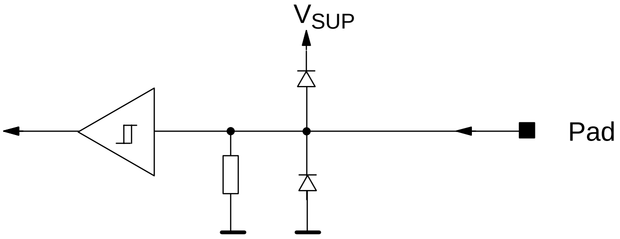


Fig. 3–8: Schmitt-Trigger input with Pull-Down Resistor

PORQTEST:	Pin 15
TEST:	Pin 40

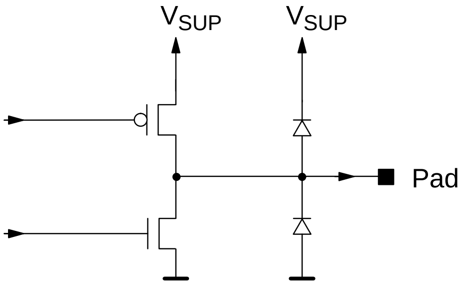


Fig. 3–5: Push-Pull output
Half-Video and Fast Blank: Pins 23, 27.

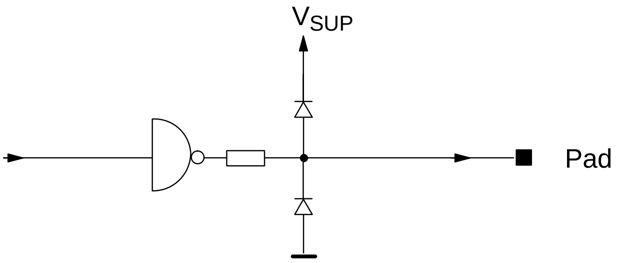


Fig. 3–6: XTAL output
XTAL2: Pin 37

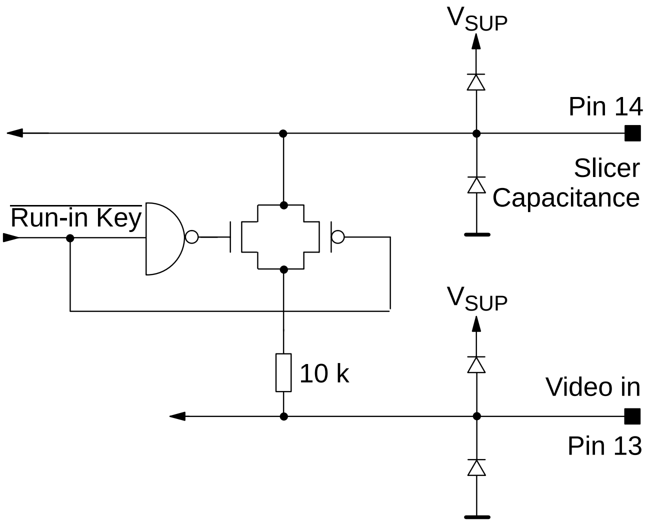


Fig. 3–9: VIDEO_{IN}, Slicer Capacitance

VIDEO_{IN}:	Pin 13
Slicer Capacitance:	Pin 14

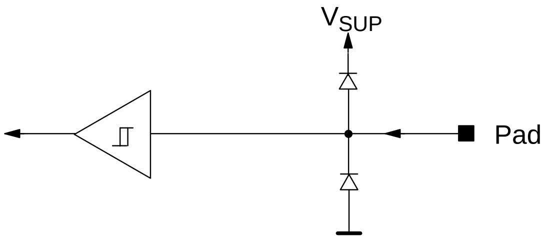


Fig. 3–10: Schmitt-Trigger input

V_{SYNC}:	Pin 29
H_{SYNC1}:	Pin 28

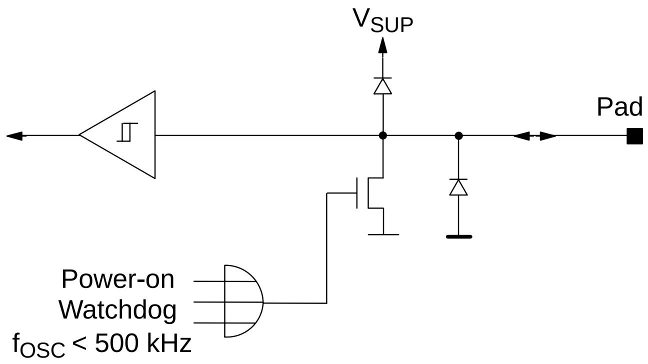


Fig. 3–11: Reset

$\overline{\text{Reset}}$:	Pin 35

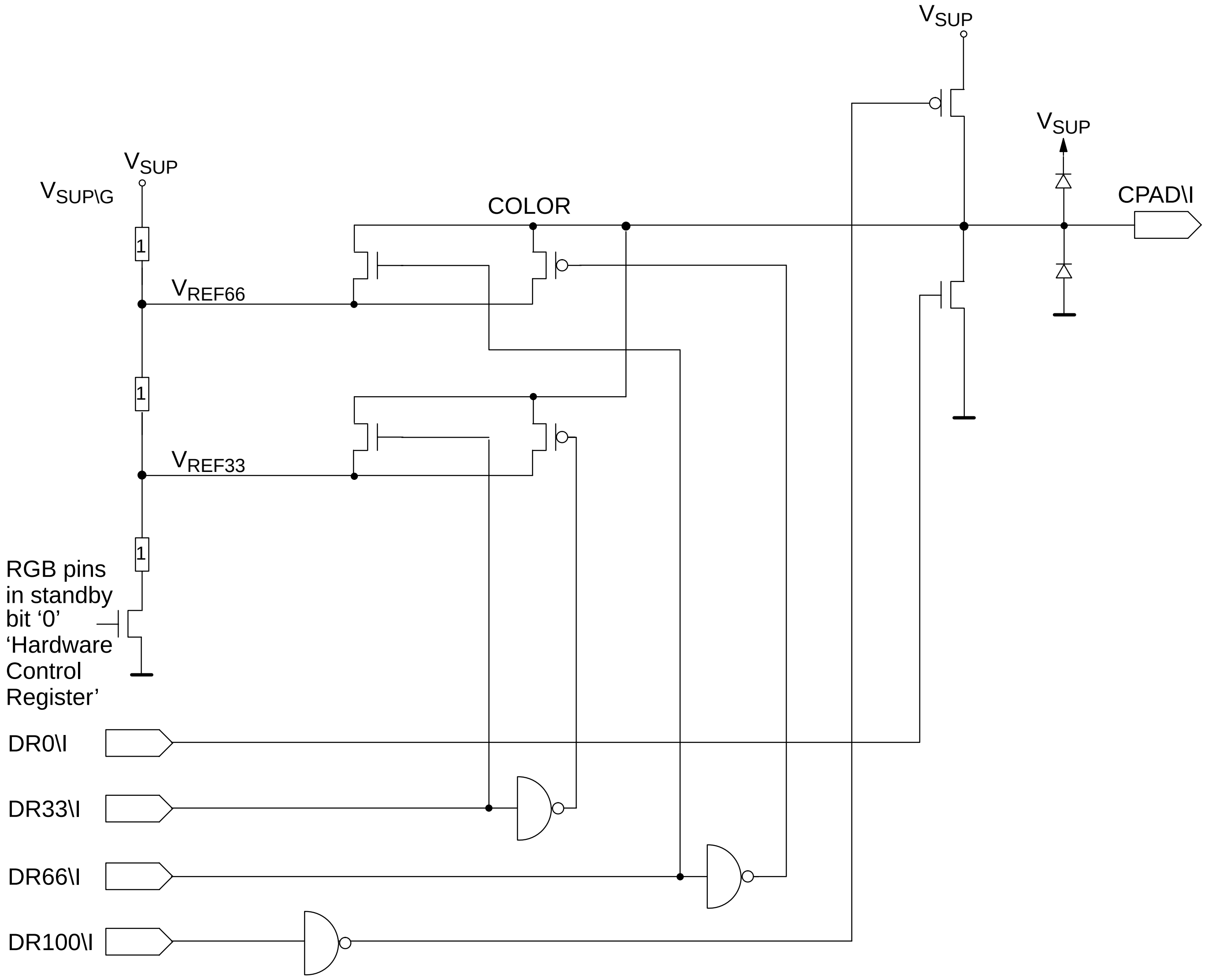


Fig. 3–12: Pin circuit RGB-out

3.6. Electrical Characteristics

All voltages refer to ground

3.6.1. Absolute Maximum Ratings

Symbol	Parameter	Pin	Min.	Max.	Unit
T_A	Ambient Operating Temperature	–	0	70	°C
T_S	Storage Temperature	–	–40	125	°C
V_{SUP}	Supply Voltage	39	–0.5	6	V
V_{SUPA}	Analog Supply Voltage	22	–0.5	6	V
I_{SUP}	Supply Current	39	–50	50	mA
I_{SUPA}	Analog Supply Current	22	–50	50	mA
V_I	Input Voltage	1 to 12, 23, 27 to 34, 41 to 52	–0.3	V_{SUP}+0.3	V
V_{IA}	Analog Input Voltage	16 to 20	–0.3	V_{SUPA} +0.3	V
I_O	Output Current	1 to 12, 23, 27 to 34, 41 to 52	–5	5	mA
$I_{OR,G,B}$	R, G, B Output Current	24 to 26	–1.6	1.6	mA
I_{OA}	Analog Output Current	13, 14, 37	–2 00	200	µA

Stresses beyond those listed in the “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only. Functional operation of the device at these or any other conditions beyond those indicated in the “Recommended Operating Conditions/Characteristics” of this specification is not implied. Exposure to absolute maximum ratings conditions for extended periods may affect device reliability.

3.6.2. Recommended Operating Conditions

at T_A = 0 to 70 °C, Fxtal = 12.0 MHz

Symbol	Parameter	Pin	Min.	Typ.	Max.	Unit
V_{SUP} 1)	Supply Voltage (digital)	39	4.75	5.0	5.25	V
V_{SUPA} 1)	Analog Supply Voltage	22	4.75	5.0	V_{SUP}	V
V_{VIDEO}	Video Input Level	13	1.0	–	2.0	V_{pp}
V_{SYNCL}	Sync Input Low Voltage	28, 29, 48	–	–	0.8	V
V_{SYNCH}	Sync Input High Voltage		3.0	–	–	V
$Xtal1_{DUTY}$	Clock Input High/Low Ratio	36	0.9	1.0	1.1	–
$Xtal1_{TRAN}$	Clock Rise/Fall Time		–	–	12.5	ns
V_{IL} 2)	Input Low Voltage	all port pins 1 to 12, 30 to 34, 41 to 52	–	–	0.8	V
V_{IH} 2)	Input High Voltage		3.0	–	–	V
RGB Analog Outputs						
R_{Load}	External Load Resistor	24 to 26	20	–	–	KΩ
C_{Load}	External Load Capacitor		–	–	20	pF
IM-Bus/I^{2}C Outputs						
R_{Load}	External Load Resistor	30 to 34	2	–	–	KΩ
C_{Load}	External Load Capacitor		–	–	100	pF
SYNC Inputs						
T_{HSYNC}	Pulse Width of Horizontal Sync	28, 48	6	–	–	T_{OSC}
T_{VSYNC}	Pulse Width of Vertical Sync	29	6	–	–	T_{OSC}

1) GND and GNDA are short-circuited on chip. Both of these pins should be hardwired to common GND plane. Capacitors should be used as near as possible to the V_{SUP} and V_{SUPA} pins against GND to minimize supply voltage disturbance. V_{SUPA} must not exceed V_{SUP}!

2) All port pins have Schmitt-Trigger inputs with a hysteresis of about 0.9 V.

3.6.3. Recommended Crystal Characteristics

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Cond.
f_{xtal}	Parallel Resonance Frequency for NTSC	11.5	12.083712	12.5	MHz	C_L = 17.5 pF
f_{xtal}	Parallel Resonance Frequency for PAL, SECAM	11.5	12.000000	12.5	MHz	C_L = 17.5 pF
$df_{p/}fp$	Frequency Deviation versus Temperature and Aging	–	–	±100	ppm	
R_r	Series Resistance	–	–	40	Ohm	C_{XTAL1} = C_{XTAL2} = 22 pF±1 pF, $C_{STRAY} \leq$ 2 pF
C_0	Shunt Capacitance	5.5	–	7	pF	
C_1	Motional Capacitance	0.015	–	0.025	pF	

3.6.4. DC Characteristics

at T_A = 0 to 70 °C, V_{SUP} = 4.75 to 5.25 V, f_{xtal} = 12 MHz for min./max. values
at T_A = 60 °C, V_{SUP} = 5 V, f_{xtal} = 12 MHz for typical values

Symbol	Parameter	Pin No.	Min.	Typ.	Max.	Unit	Test Conditions
I_{SUP}[1]	Supply Current (digital)	39	–	22	50	mA	no loads on outputs
I_{SUPA}[1]	Analog Supply Current	22	–	–	1	mA	no loads on outputs
$I_{SUPASTBY}$[1] [2]	Analog Standby Supply Current		–	–	1	µA	no loads on outputs
ADC Inputs							
I_{LIA}[1]	Analog Input Leakage Current	16 to 20	–1	–	1	µA	GND≤V_{IN}≤V_{SUP}
Inputs							
I_{LI}	Leakage Current	1 to 14, 23, 27 to 35, 41 to 52	–1	–	1	µA	GND≤V_{IN}≤V_{SUP}
RGB Analog Outputs							
R_i	Internal Resistance	24 to 26	–	–	2	KΩ	(guaranteed by design)
Port Outputs							
V_{OL}	Output Low Voltage	1 to 12, 30 to 34, 41 to 52			0.4	V	I_{OUT} = 2 mA
V_{OH}	Output High Voltage		V_{SUP}–0.5			V	I_{OUT} = –1 mA
Port 1 in Special Mode (I^2C-Bus)							
V_{OD}	Output Low Voltage in Open Drain Configuration	30, 31, 33, 34			0.4	V	I_{OUT} = 5 mA
Reset Pin							
V_{ROL}	Reset Pin Output Low Voltage	35			0.4	V	I_{OUT} = 2 mA

[1] GND and GNDA are short-circuited on chip. Both of these pins should be hardwired to common GND plane. Capacitors should be used as near as possible to the V_{SUP} and V_{SUPA} pins against GND to minimize supply voltage disturbance. V_{SUPA} must not exceed V_{SUP}!

[2] ADC hardware disabled with bit 3 in 'Hardware Control' register, address 209H, set to '0'.

3.6.5. DC Parameters I^{2}C-Bus Master Interface

The input and output parameters of the I^{2}C-bus interface (Clock and Data) are designed according to the Micronas specification for Port and IM-bus pins (the interface can also be operated as IM-bus interface). The differences are

Symbol	Meaning	Micronas	I^{2}C Specification
V_{IL}	Input Low Voltage	max. 0.8 V	max. 1.5 V
V_{IH}	Input High Voltage	min. 2.5 V	min. 3 V
V_{OL}	Output Low Voltage	0.4 / 5 mA	0.4 V / 3 mA

The Micronas parameters are equivalent to software I^{2}C-bus solutions using Port-lines for the bus. In applications with series resistors in the clock or data line these differences may become important. Capacities on any of these pins should not exceed 100 pF. Higher capacities could effect higher disturbances.

Table 2–9: AC Characteristics

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
T_{PH2}	CPU Cycle Time	–	2	–	T_{OSC}	–

3.6.6. A/D Converter Characteristics

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
–	Resolution	–	–	8	Bits	
–	Absolute Accuracy	–	–	±4	LSB	
t_{CONV}	Conversion Time	–	–	68	T_{OSC}	
V_{IA}	Analog Input Voltage	–	–	V_{SUPA}	V	
DW_{MIN}	Minimum Digital Value	00H	–	–		$V_{IA}<V_{SUPA}$
DW_{MAX}	Maximum Digital Value	–	–	FFH		$V_{IA}\geq V_{SUPA}$

4. Definitions

4.1. I/O Definitions

Address	Function
200H	System Clock Prescaler Register
201H	CCZ Control Register
202H	Watchdog Control and Status Register
203H	System Counter LS Byte
204H	System Counter MS Byte
209H	Hardware Control Register
236H	NMI Return Register
237H	Window Logic Control Register 2
238H	Window Logic Control Register 1
239H	Sync Tip Clamp Start Value Register
23AH	Sync Tip Clamp End Value Register
23BH	Run-In Key Start Value Register
23CH	Run-In Key End Value Register
23DH	Vertical Sync Phase Value Register (field 1/field 2 determination)
23EH	Closed Caption Line Number
23FH	Captured Data Register
250H	PWM 0 Data Register
251H	PWM 1 Data Register
252H	PWM 2 Data Register
253H	PWM 3 Data Register
254H	PWM 4 Data Register
255H	PWM 5 Data Register
256H	PWM 6 Data Low Register
257H	PWM 6 Data High Register
260H	OSD First Active Scan Line (9 bits)
261H	OSD Last Active Scan Line (9 bits)
262H	OSD Horizontal Start Position
263H	OSD Control Register
264H	OSD Text Start Address Register
265H	OSD Separate Colored Line (9 bits)
266H	OSD Separate Color Definition Register
267H	OSD Color Palette red
268H	OSD Color Palette green
269H	OSD Color Palette blue
26AH	OSD Half-Video Control Register
26EH	OSD Font1, Font2 Start Addresses (32 bits)
26FH	OSD Horizontal Start Adjust and Display Options
270H:	Cursor Pointer, low byte
271H:	Cursor Pointer, high byte
272H:	Cursor X–Position, low byte
273H:	Cursor X–Position, high byte
274H:	Cursor X–Position, low byte
275H:	Cursor X–Position, high byte
276H:	Field Detector OSD
277H	Horizontal Sync Tracking
278H	Vertical Sync Tracking
279H	H&V Sync Generator Mode
280H	Port 0 Mode Register
281H	Port 0 Tristate Register
282H	Port 0 Data Register
284H	Port 1 Mode Register
285H	Port 1 Tristate Register
286 H	Port 1 Data Register
288H	Port 2 Mode Register
289H	Port 2 Tristate Register
28AH	Port 2 Data Register
28CH	Port 3 Mode Register
28DH	Port 3 Tristate Register
28EH	Port 3 Data Register
2A8H	Analog Input Select and Status Register
2A9H	Analog Value Register
2D0H	I^{2}C Start Cycle without generation of ACK (ACK = ‘1’)
2D1H	I^{2}C Start Cycle with generation of ACK (ACK = ‘0’)
2D2H	I^{2}C Resume Cycle without generation of ACK (ACK = ‘1’)
2D3H	I^{2}C Resume Cycle with generation of ACK (ACK = ‘0’)
2D4H	I^{2}C End Cycle without generation of ACK (ACK = ‘1’)
2D5H	I^{2}C Termination Cycle with generation of ACK (ACK = ‘0’)
2D6H	I^{2}C / IM-bus Data from Receive-FIFO
2D7H	I^{2}C / IM-bus Status
2D8H	IM-bus Start Cycle
2D9H	IM-bus Resume Cycle
2DAH	IM-bus End Cycle
2DBH	I^{2}C / IM-bus Prescaler
2E0H to 2E7H	External Hardware Access (used for emulation purposes)
2F5H	IR Control and Status Register
2F6H	IR Sample Times Register
2FEH	Test Register 1 (don’t use, reserved for test purposes)
2FFH	Test Register 2 (don’t use, reserved for test purposes)

5. Register Description

(x corresponds to undefined for READ, no function for WRITE.) Do not access any other addresses in page 2 than mentioned here. Fill unused register bits with ‘0’ to be software-compatible with other versions of CCZ 3005K.

0200H	**System Clock Prescaler Register (use not recommended, as not tested)**		
Bit	**Reset**	**Read**	**Write**
7 to 0	0	x	Divisor value –1, determines f_{system}: $f_{system} = f_{XTAL}$ / 2* (Divisor value+1)

0201H	**CCZ Control Register**			
Bit	**Reset**	**Read**	**Write**	
7 to 3		x	1	copied from addr. FFF9H during RESET
2		‘0’ = ROM external , ‘1’ = ROM internal	‘0’ = ROM external , ‘1’ = ROM internal	
1		‘0’ = RAM external , ‘1’ = RAM internal	‘0’ = RAM external , ‘1’ = RAM internal	
0		‘0’ = CPU external , ‘1’ = CPU internal	‘0’ = CPU external , ‘1’ = CPU internal	

0202H	**Watchdog Control and Status Register**		
Bit	**Reset**	**Read**	**Write**
7	x	x	$\frac{f_{system}}{65536} \cdot T_{WD} - 1 = n$ with f_{system} = 6 MHz: $n = n_{min} = 2 \Rightarrow T_{WDmin} = 32.768$ ms $n = n_{max} = 255 \Rightarrow T_{WDmax} = 2.796$ s min. Δ n = 1 $\Rightarrow$ min. $\Delta T_{WD} = 10.922666$ ms *(do not use settings n < 2!!)* *(the watchdog is disabled after RESET)*
6	x	x	
5	x	x	
4	x	x	
3	x	x	
2	x	x	
1	x	Test	
0	x	‘0’ = RESET was generated by watchdog	

0203H	**System Counter LS Byte**		
Bit	**Reset**	**Read**	**Write**
7	0	Bit 7	x
6	0		x
5	0		x
4	0	Counter value low byte	x
3	0		x
2	0		x
1	0		x
0	0	Bit 0	x

0204H	**System Counter MS Byte**		
Bit	**Reset**	**Read**	**Write**
7	0	Bit 15	x
6	0		x
5	0		x
4	0	Counter value high byte *(latched by reading counter value low byte)*	x
3	0		x
2	0		x
1	0		x
0	0	Bit 8	x

0209H	**Hardware Control Register**		
Bit	**Reset**	**Read**	**Write**
7 to 6	x	x	x
5	0	x	port buffers, Half Video and Fast Blank output control: ‘0’ = normal, ‘1’ = current-controlled
4	x	x	x
3	0	x	‘1’ = enable ADC hardware, ‘0’ = ADC hardware in standby
2 to 1	x	x	x
0	0	x	‘1’ = RGB hardware in standby, output pins tristate

0236H	**NMI-Return Register**		
Bit	**Reset**	**Read**	**Write**
7 to 0	x	x	x enables the succeeding NMI with the next processor SYNC signal *(NMIs automatically become disabled with occurrence.)*

0237H	**Window Logic Control Register 2**		
Bit	**Reset**	**Read**	**Write**
7 to 4	x	x	x
3	0	selected Hsync input: ‘0’: H_{SYNC1}, ‘1’: H_{SYNC2}	Hsync input select: ‘0’: H_{SYNC1}, ‘1’: H_{SYNC2}
2	0	‘0’ = video peak clamp, ‘1’ = video gated clamp	‘0’ = video peak clamp, ‘1’ = video gated clamp
1	0	detect video: ‘0’ = Run-In Key enabled, ‘1’ = Run-In Key disabled	detect video: ‘0’ = enable Run-In Key, ‘1’ = disable Run-In Key
0	0	‘0’ = 3 line interrupt disabled, ‘1=’ 3 line interrupt enabled	‘0’ = disable 3 line interrupt, ‘1’ = enable 3 line interrupt

0238H	Window Logic Control Register 1		
bits	**Reset**	**Read**	**Write**
7	x	‘0’: Vsync inactive, ‘1’: Vsync active	x
6	1	‘0’ = line counter inactive, ‘1’ = line counter active	‘0’ = deactivate line counter, ‘1’ = activate line counter
5	0	‘0’ = CPU_$\overline{\text{SO}}$ input disabled, ‘1’ = CPU_$\overline{\text{SO}}$ input enabled	‘0’ = disable CPU_$\overline{\text{SO}}$ input , ‘1’ = enable CPU_$\overline{\text{SO}}$ input
4	1	‘0’ = halfdot rounding disabled, ‘1’ = halfdot rounding enabled	‘0’ = disable halfdot rounding, ‘1’ = enable halfdot rounding[1]
3	x	‘0’ = OSD active , ‘1’ = OSD inactive	x
2	1	active edge of Vsync: ‘0’: rising, ‘1’: falling	active edge of Vsync: ‘0’: rising, ‘1’: falling
1	1	active edge of Hsync for acquisition: ‘0’: rising, ‘1’: falling	active edge of Hsync for acquisition: ‘0’: rising, ‘1’: falling
0	1	active edge of Hsync for OSD: ‘1’: rising, ‘0’: falling	active edge of Hsync for OSD: ‘1’: rising, ‘0’: falling

[1] Halfdot rounding may be active during field 2 (even field) only, so software has to toggle this bit with every active V_{sync} edge. Halfdot rounding not defined for Horizontal Start position (Reg. 0262H) less than 6 and greater than 46.

0239H	Sync Tip Clamp Start Value Register		
Bit	**Reset**	**Read**	**Write**
7 to 0	1	synctip start value	synctip start value

023AH	Sync Tip Clamp End Value Register		
Bit	**Reset**	**Read**	**Write**
7 to 0	1	synctip stop value (duration of synctip: = (reg23A – reg239)*4/PXTAL)	synctip stop value (duration of synctip: = (reg23A – reg239)*4/PXTAL)

023BH	Run-in Key Start Value Register		
Bit	**Reset**	**Read**	**Write**
7 to 0	1	run-in key start value	run-in key start value

023CH	Run-in Key End Value Register		
Bit	**Reset**	**Read**	**Write**
7 to 0	1	run-in key stop value (duration of run-in key: = (reg23C – reg239B)*4/PXTAL)	run-in key stop value (duration of run-in key: = (reg23C – reg239B)*4/PXTAL)

023DH	Vertical Sync Phase Value Register (field 1/field 2 determination)		
Bit	**Reset**	**Read**	**Write**
7 to 0	x	vertical sync phase value	x

023EH	Closed Caption Line Number		
Bit	**Reset**	**Read**	**Write**
7 to 0	1	number of closed caption line	number of closed caption line (both V_{sync} and caption line interrupt (NMI) is-disabled until a first write to this register occurs).

023FH	Captured Data Register		
Bit	**Reset**	**Read**	**Write**
7 to 0	x	captured data	x

0250H	PWM0 Data Register		
Bit	**Reset**	**Read**	**Write**
7	0	x	x
6	0	x	x
5 to 0	0	x	PWM counter value

0251H	PWM1 Data Register		
Bit	**Reset**	**Read**	**Write**
7	0	x	x
6	0	x	x
5 to 0	0	x	PWM counter value

0252H	**PWM2 Data Register**		
Bit	**Reset**	**Read**	**Write**
7	0	x	x
6	0	x	x
5 to 0	0	x	PWM counter value

0253H	**PWM3 Data Register**		
Bit	**Reset**	**Read**	**Write**
7	0	x	x
6	0	x	x
5 to 0	0	x	PWM counter value

0254H	**PWM4 Data Register**		
Bit	**Reset**	**Read**	**Write**
7	0	x	x
6	0	x	x
5 to 0	0	x	PWM counter value

0255H	**PWM5 Data Register**		
Bit	**Reset**	**Read**	**Write**
7	0	x	x
6	0	x	x
5 to 0	0	x	PWM counter value

0256H	**PWM6 Data Low Register**		
Bit	**Reset**	**Read**	**Write**
7	0	x	x
6	0	x	x
5 to 0	0	x	PWM counter low value

0257H	**PWM6 Data High Register**		
Bit	**Reset**	**Read**	**Write**
7 to 0	0	x	PWM counter high value *(writing this register transfers the complete 14–bit value into the PWM6 counter)*

0260H	**OSD First Active Scan Line (9 bits) of OSD Window (Set high bits to ‘1’)** (vertical start of OSD window)		
Bit	**Reset**	**Read**	**Write**
7 to 0	0	x	Number of the first displayed OSD-scan-line *(the first access defines the high bit (bit 0), the second access defines the low byte of the number; two accesses are always required)*

0261H	**OSD Last Active Scan Line (9 bits) of OSD Window (Set high bits to ‘1’)** (vertical stop of OSD window)		
Bit	**Reset**	**Read**	**Write**
7 to 0	0	x	Number of the last displayed OSD-scan line *(the first access defines the high bit (bit 0), the second access defines the low byte of the number; two accesses are always required)*

0262H	**OSD Horizontal Start Position of OSD Window (in character steps)**		
Bit	**Reset**	**Read**	**Write**
7	0	x	x
6	0	x	x
5 to 0	0	x	OSD-x-start position (must be greater than 1) (don’t use values less than 6 and greater than 46 if halfdot rounding is enabled (bit 4 in register 0238H = window logic controlreg_1 set to ‘1’)

0263H	**OSD Control Register**		
Bit	**Reset**	**Read**	**Write**
7	0	x	‘1’ = enable NCI Decoder (telecaption mode of the OSD), ‘0’ = enable standard OSD mode
6	0	x	‘1’ = display on ‘0’ = display off
5	0	x	‘0’ = flash on, ‘1’ = flash off (foreground= background)
4	0	x	‘0’ = font size 13x8 (NTSC), ‘1’ = 15x8 (PAL)
3 to 0	0	x	Start scan line: selects the first scan line of the first character row. This capability allows the smooth scrolling feature to look correct at the top of the window.

0264H	**OSD Text Start Address Register**		
Bit	**Reset**	**Read**	**Write**
7 to 0	0	x	Start address of text to be displayed *(the first access defines the high byte, the second access defines the low byte of the address; two accesses are always required)*

0265H	**OSD Separate Colored Line (9 bits) (Set high bits to ‘1’)**		
Bit	**Reset**	**Read**	**Write**
7 to 0	0	x	Number of second color scan line *(the first access defines the high bit, (bit 0); the second access defines the low byte of the number; two accesses are always required)*

0266H	**OSD Separate Color Definition Register (Bits 0 to 6: same format as attribute ‘color’)**		
Bit	**Reset**	**Read**	**Write**
7	0	x	‘0’ = fastblank active low, ‘1’ = fastblank active high
6	0	x	‘1’= color 0 is replaced by transparent
5 to 3	0	x	value 0 to 7 defining background color (color 0 to color 7)
2 to 0	0	x	value 0 to 7 defining foreground color (color 0 to color 7)

0267H	**OSD Color Palette red**			
Bit	**Reset** **high** **(2nd access)**	 **low** **(1st access)**	**Read**	**Write**
7	1	1	x	Color 7
6	1	1	x	Color 6
5	1	1	x	Color 5 *(the first access defines the low bits, the second access defines the high bits of the colors intensity, two accesses are always required)*
4	1	1	x	Color 4
3	0	0	x	Color 3
2	0	0	x	Color 2
1	0	0	x	Color 1
0	0	0	x	Color 0

0268H	**OSD Color Palette green**			
Bit	**Reset** **high** **(2nd access)**	 **low** **(1st access)**	**Read**	**Write**
7	1	1	x	Color 7
6	1	1	x	Color 6
5	0	0	x	Color 5 *(the first access defines the low bits, the second access defines the high bits of the colors intensity, two accesses are always required)*
4	0	0	x	Color 4
3	1	1	x	Color 3
2	1	1	x	Color 2
1	0	0	x	Color 1
0	0	0	x	Color 0

0269H	**OSD Color Palette blue**			
Bit	**Reset** **high** **(2nd access)**	 **low** **(1st access)**	**Read**	**Write**
7	1	1	x	Color 7
6	0	0	x	Color 6
5	1	*(the first access defines the low bits, the second access defines the high bits of the colors intensity, two accesses are always required)*	x	Color 5
4	0		x	Color 4
3	1		x	Color 3
2	0		x	Color 2
1	1	1	x	Color 1
0	0	0	x	Color 0

026AH	OSD Half-Video Control Register		
Bit	**Reset**	**Read**	**Write**
7 to 1	x	x	x
0	0	x	‘0’= disable half-video (half-video output pin level=‘0’) ‘1’=enable half-video

026EH	OSD FONT1, FONT2 Start Addresses (32 bits)		
Bit	**Reset**	**Read**	**Write**
7	0	x	Start address of FONT1 and FONT2 *(the first access defines the high byte, the second access the low byte of the FONT1 address; the third access defines the high byte, the fourth access the low byte of the FONT2 address; there are always 4 accesses required!)*
6	0	x	
5	0	x	
4	0	x	
3	0	x	
2	0	x	
1	0	x	
0	0	x	

026FH	OSD Horizontal Start Adjust and Display Options		
Bit	**Reset**	**Read**	**Write**
7	0	x	enable horizontal shadow
6	0	x	blank display
5	0	x	‘0’ = Display Mode 0 ‘1’ = Display Mode 1
4	0	x	‘0’ = single underline in scan line 11 ‘1’ = single underline in scan line 12 (works with 13 x 8 font only)
3	x	x	x
2	1	x	x-fine adjust
1	1	x	
0	1	x	

0270H	Cursor Pointer, low byte		
Bit	**Reset**	**Read**	**Write**
7 to 0	0	x	Start address of the cursor definition bit-map (low one of two bytes)

0271H	**Cursor Pointer, high byte**		
Bit	**Reset**	**Read**	**Write**
7 to 0	0	x	Start address of the cursor definition bit-map (high one of two bytes)

0272H	**Cursor X–Position, low byte**		
Bit	**Reset**	**Read**	**Write**
7 to 0	1	X–position (horizontal start) of cursor (least significant eight of nine bits)	X–position (horizontal start) of cursor (least significant eight of nine bits)

0273H	**Cursor X–Position, high byte**		
Bit	**Reset**	**Read**	**Write**
7	0	Cursor on/off–switch: ’0’: Cursor off ’1’: Cursor on	Cursor on/off–switch: ‘0’: Cursor off ‘1’: Cursor on
6 to 4	0	Cursor color mask 2	Cursor color mask 2
3 to 1	0	Cursor color mask 1	Cursor color mask 1
0	1	X–position (horizontal start) of cursor (most significant of nine bits)	X–position (horizontal start) of cursor (most significant of nine bits)

0274H	**Cursor Y–Position, low byte**		
Bit	**Reset**	**Read**	**Write**
7 to 0	0	Y–position (vertical start) of cursor (least significant eight of nine bits)	Y–position (vertical start) of cursor (least significant eight of nine bits)

0275H	**Cursor Y–Position, high byte**		
Bit	**Reset**	**Read**	**Write**
7	0	Cursor transparent–switch: ‘0’: transparent off ‘1’: transparent on	Cursor transparent–switch: ‘0’: transparent off ‘1’: transparent on
6	0	Cursor half–video–switch: ’0’: half video off ’1’: half video on	Cursor half–video–switch: ‘0’: half video off ‘1’: half video on
5	0	Cursor fast blank polarity: ’0’: active low ’1’: active high	Cursor fast blank polarity: ‘0’: active low ‘1’: active high
4	x	x	x
3 to 1	0	Cursor color mask 3	Cursor color mask 3
0	1	Y–position (vertical start) of cursor (most significant of nine bits)	Y–position (vertical start) of cursor (most significant of nine bits)

0276H	**Field Detector OSD (field 1/field 2 determination)**		
Bit	**Reset**	**Read**	**Write**
7 to 0	x	vertical sync phase value	x

0277H	**Horizontal Sync Tracking Speed**		
Bit	**Reset**	**Read**	**Write**
7 to 5	0	x	x
4 to 0	0	x	tracking speed for horizontal sync timer

0278H	**Vertical Sync Tracking Speed**		
Bit	**Reset**	**Read**	**Write**
7 to 5	0	x	x
4 to 0	0	x	tracking speed for vertical sync timer

0279H	**H&V Sync Generator Mode**		
Bit	**Reset**	**Read**	**Write**
7	0	x	x
6	0	x	‘0’ = normal V_{OSD} output ‘1’ = inverted V_{OSD} output
5	0	x	‘0’ = normal H_{OSD} output ‘1’ = inverted H_{OSD} output
4	0	x	‘0’ = vertical tracking disabled ‘1’ = vertical tracking enabled
3	0	x	‘0’ = horizontal tracking disabled ‘1’ = horizontal tracking enabled
2	0	x	‘0’ = separate $V_{OSD\ \&}$ H_{OSD} output ‘1’ = composite sync output
1	0	x	‘0’ = pal mode ‘1’ = ntsc mode
0	1	x	‘0’ = selftimed sync mode ‘1’ = external sync mode

0280H	**Port0 Mode Register**		
Bit	**Reset**	**Read**	**Write**
7	0	x	‘0’ = normal mode, ‘1’ = special mode
6	0	x	‘0’ = normal mode, ‘1’ = special mode
5	0	x	‘0’ = normal mode, ‘1’ = special mode (H_{OSD} output)
4	0	x	‘0’ = normal mode, ‘1’ = special mode (V_{OSD} output)
3	0	x	‘0’ = normal mode, ‘1’ = special mode
2	0	x	‘0’ = normal mode, ‘1’ = special mode
1	0	x	‘0’ = normal mode, ‘1’ = special mode
0	0	x	‘0’ = normal mode, ‘1’ = special mode

0281H	**Port0 Tristate Register**		
Bit	**Reset**	**Read**	**Write**
7 to 0	1	x	‘0’ = output conducting, ‘1’ = output tristate

0282H	**Port0 Input Data/Output Register**		
Bit	**Reset**	**Read**	**Write**
7 to 0	0	Port0 data	Port0 data

0284H	**Port1 Mode Register**		
Bit	**Reset**	**Read**	**Write**
7 to 5	x	x (not available)	x (not available)
4	0	x	‘0’ = normal mode, ‘1’ = special mode (IM_1-CLK/I^2C_1-SCL)
3	0	x	‘0’ = normal mode, ‘1’ = special mode (IM_1-DAT/I^2C_1-SDA)
2	0	x	‘0’ = normal mode, ‘1’ = special mode (IM-ID)
1	0	x	‘0’ = normal mode, ‘1’ = special mode (IM_2-CLK/I^2C_2-SCL)
0	0	x	‘0’ = normal mode, ‘1’ = special mode (IM_2-DAT/I^2C_2-SDA)

0285H	**Port1 Tristate Register (only 5 bits available)**		
Bit	**Reset**	**Read**	**Write**
4 to 0	1	x	‘0’ = output conducting, ‘1’ = output tristate

0286H	**Port1 Data Register (only 5 bits available)**		
Bit	**Reset**	**Read**	**Write**
4 to 0	0	Port1 data	Port1 data

0288H	**Port2 Mode Register**		
Bit	**Reset**	**Read**	**Write**
7	0	x	‘0’ = normal mode, ‘1’ = special mode (H_{SYNC2} input)
6	0	x	‘0’ = normal mode, ‘1’ = special mode (Ext. Slicer)
5	0	x	‘0’ = normal mode, ‘1’ = special mode (PWM5 output)
4	0	x	‘0’ = normal mode, ‘1’ = special mode (PWM4 output)
3	0	x	‘0’ = normal mode, ‘1’ = special mode (PWM3 output)
2	0	x	‘0’ = normal mode, ‘1’ = special mode (PWM2 output)
1	0	x	‘0’ = normal mode, ‘1’ = special mode (PWM1 output)
0	0	x	‘0’ = normal mode, ‘1’ = special mode (PWM0 output)

0289H	**Port2 Tristate Register**		
Bit	**Reset**	**Read**	**Write**
7 to 0	1	x	‘0’ = output conducting, ‘1’ = output tristate

028A	**Port2 Data Register**		
Bit	**Reset**	**Read**	**Write**
7 to 0	0	Port2 data	Port2 data

028CH	**Port3 Mode Register**		
Bit	**Reset**	**Read**	**Write**
7	0	x	‘0’ = normal mode, ‘1’ = special mode (PWM6 output)
6	0	x	‘0’ = normal mode, ‘1’ = special mode (IR_{IN} output)
5	0	x	‘0’ = normal mode, ‘1’ = special mode (IR_{IN})
4	0	x	‘0’ = normal mode, ‘1’ = special mode (A2 output)
3	0	x	‘0’ = normal mode, ‘1’ = special mode (CC Line)
2	0	x	‘0’ = normal mode, ‘1’ = special mode (Synctip Clamp Gate)
1	0	x	‘0’ = normal mode, ‘1’ = special mode ($\overline{\text{Run-In Key}}$)
0	0	x	‘0’ = normal mode, ‘1’ = special mode (Average)

028DH	**Port3 Tristate Register**		
Bit	**Reset**	**Read**	**Write**
7 to 0	1	x	‘0’ = output conducting, ‘1’ = output tristate

028EH	**Port3 Data Register**		
Bit	**Reset**	**Read**	**Write**
7 to 0	0	Port3 data	Port3 data

02A8H	**Analog Input Select and Status Register**		
Bit	**Reset**	**Read**	**Write (A/D conversion is started)**
7	0	‘EOC’-flag, if ‘1’: A/D conversion terminated	x
6 to 3	x	x	x
2 to 0	0	x	No. of analog input pin to convert voltage from: 0 to 4 for ADC0 to ADC4

02A9H	**A/D Converter Output Value Register**		
Bit	**Reset**	**Read**	**Write**
7 to 0	x	Analog value converted from selected ADC pin. Only valid after ‘EOC’ = ‘1’	x

02D0H	**I^{2}C Start Cycle without Generation of ACK (ACK=‘1’)**		
Bit	**Reset**	**Read**	**Write**
7 to 0	x	x	I^{2}C-Start-Data

02D1H	**I^{2}C Start Cycle with Generation of ACK (ACK=‘0’)**		
Bit	**Reset**	**Read**	**Write**
7 to 0	x	x	I^{2}C-Start-Data

02D2H	**I^{2}C Resume Cycle without Generation of ACK (ACK=‘1’)**		
Bit	**Reset**	**Read**	**Write**
7 to 0	x	x	I^{2}C-Resume-Data *(set this byte to $FF for a read access)*

02D3H	**I^{2}C Resume Cycle with Generation of ACK (ACK=‘0’)**		
Bit	**Reset**	**Read**	**Write**
7 to 0	x	x	I^{2}C-Resume-Data *(set this byte to $FF for a read access)*

02D4H	**I^{2}C End Cycle without Generation of ACK (ACK=‘1’)**		
Bit	**Reset**	**Read**	**Write**
7 to 0	x	x	I^{2}C-Terminate-Data *(set this byte to $FF for a read access)*

02D5H	**I^{2}C End Cycle with Generation of ACK (ACK=‘0’)**		
Bit	**Reset**	**Read**	**Write**
7 to 0	x	x	I^{2}C-Terminate-Data *(set this byte to $FF for a read access)*

02D6H	**I^{2}C/IM-bus Data from Receive-FIFO**		
Bit	**Reset**	**Read**	**Write**
7 to 0	x	Received data	x

02D7H	I^2C/IM-bus Status		
Bit	**Reset**	**Read**	**Write**
7	0	x	x
6	0	I^2C OR'D ACK	x
5	0	I^2C ADDR – ACK	x
4	0	I^2C Data – ACK	x
3	0	Bus busy	x
2	0	WR FIFO half full	x
1	0	RD FIFO empty	x
0	0	x	x

02D8H	IM-bus Start Cycle		
Bit	**Reset**	**Read**	**Write**
7 to 0	x	x	IM-bus start – (address) – data

02D9H	IM-bus Resume Cycle		
Bit	**Reset**	**Read**	**Write**
7 to 0	x	x	IM-bus resume data

02DAH	IM-bus End Cycle		
Bit	**Reset**	**Read**	**Write**
7 to 0	x	x	IM-bus terminal data

02DBH	I^2C/IM-bus Prescaler		
Bit	**Reset**	**Read**	**Write**
7	1	x	'0' = select IM/I^2C-bus2, '1' = select IM/I^2C-bus1

6	0	x	$\frac{f_{XTAL}}{8 \cdot \text{bit rate}} = n$
			for $n \neq 0$: bit rate= $\frac{f_{XTAL}}{8 \cdot n}$
			for n=0: bit rate= $\frac{f_{XTAL}}{8 \cdot 128}$

5	0	x	
4	0	x	
3	0	x	
2	0	x	
1	0	x	
0	0	x	

02F5H	**IR Control and Status (Bits 0, 1, 3 and 4 are cleared after read)** (Writing into this register may generate an IRQ).		
Bit	**Reset**	**Read**	**Write**
7	0	x	Input Trigger edge: ‘0’ = falling, ‘1’ = rising
6	0	IR_{IN}-pin level at moment T_2	Infrared IRQ: ‘0’ = disabled, ‘1’ = enabled
5	0	x	IR_{IN} sample rate: ‘0’ = 85 µs, ‘1’ = 170 µs
4	0	If ‘1’: timer caused interrupt	x
3	0	If ‘1’: IR_{IN} active start edge found	IRQ source selection: ‘0’: active IR_{IN} start edge ‘1’: both IR_{IN} samples taken
2	0	IR_{IN} level (direct, as if read by standard input port)	x
1	0	If ‘1’: both IR_{IN} samples taken	x
0	0	IR_{IN}-pin level at moment T_1	x

02F6H	**IR Sample Times Register**		
Bit	**Reset**	**Read**	**Write**
7 to 4	2	x	IR_{IN} sample time 2: T_2 (0 to 14, 15 is not allowed)
3 to 0	1	x	IR_{IN} sample time 1: T_1 (0 to 14, 15 is not allowed)

02FEH	**Test Register 1 (don’t use, reserved for test purposes)**		
Bit	**Reset**	**Read**	**Write**
7 to 0			

02FFH	**Test Register 2 (don’t use, reserved for test purposes)**		
Bit	**Reset**	**Read**	**Write**
7 to 0			

6. Appendix A: Closed Caption

This chapter comprises parts of the standard “Recommended Practice for Line 21 Data Service” published as EIA-608 by the Electronic Industries Association, Washington, USA.

The CCZ 3005K is intended for the use in conventional analog or digital television receivers. The CCZ 3005K is able to decode Closed Caption transmissions in the NTSC, PAL, and Secam TV standards (PAL and Secam standards provided the Closed Caption data is transmitted in the same format as shown in Fig. 6–1 with a bit data rate of ±5% of 503 KHz). The IC contains a very powerful on-screen display (OSD) facility which can handle all of the TV’s display functions, as well as displaying the Closed Caption data. The CCD 3000 may be used in set-top decoder boxes with the addition of a sync processor.

6.1. The Closed Caption Standard

6.1.1. Data Transmission Format

Captions associated with a television program are transmitted as an encoded composite data signal during line 21 of field one of the standard NTSC video signal as shown in Fig. 6–1. The signal consists of a clock run-in signal, a start-bit, and 16 bits of data corresponding to two separate bytes of 8 bits each including parity. Therefore, transmission of actual data amounts to 16 bits every 1/30th of a second or 480 bits per second. This data stream contains encoded information which provides the instructions for display formatting and the characters to be displayed. The clock run-in consists of a seven cycle sinusoidal burst which is frequency and phase locked to the caption data clock rate. The clock run-in signal, whose frequency (32 f_h) is twice that of the data, can be used to provide synchronization for the decoder clock. The clock run-in signal is followed by two data bits at a ‘0’ logical level, then by a logical ‘1’ start-bit.

The transmitted data is coded in a non-return-to-zero (NRZ) format. All control and alpha-numeric characters utilize the seven-bit code of the US Standard Code for Information Interchange (USASCII). An eighth bit is added to each character to provide odd parity for error detection.

The sequence of identification, control, and character transmission is shown in Figs. 6–2 through 6–4. Each caption transmission is preceded by a preamble control code which consists of a non-printing character and a printing character to form a row address and display color code. Both characters of the preamble control code and all control codes are always transmitted in a single line 21 and are transmitted twice in succession to ensure correct reception of control information. Transmitted caption may be interrupted by mid-caption control codes between two complete words, in order to change display conditions, such as color or italics. At the completion of a caption transmission, an end-of-caption control code is sent.

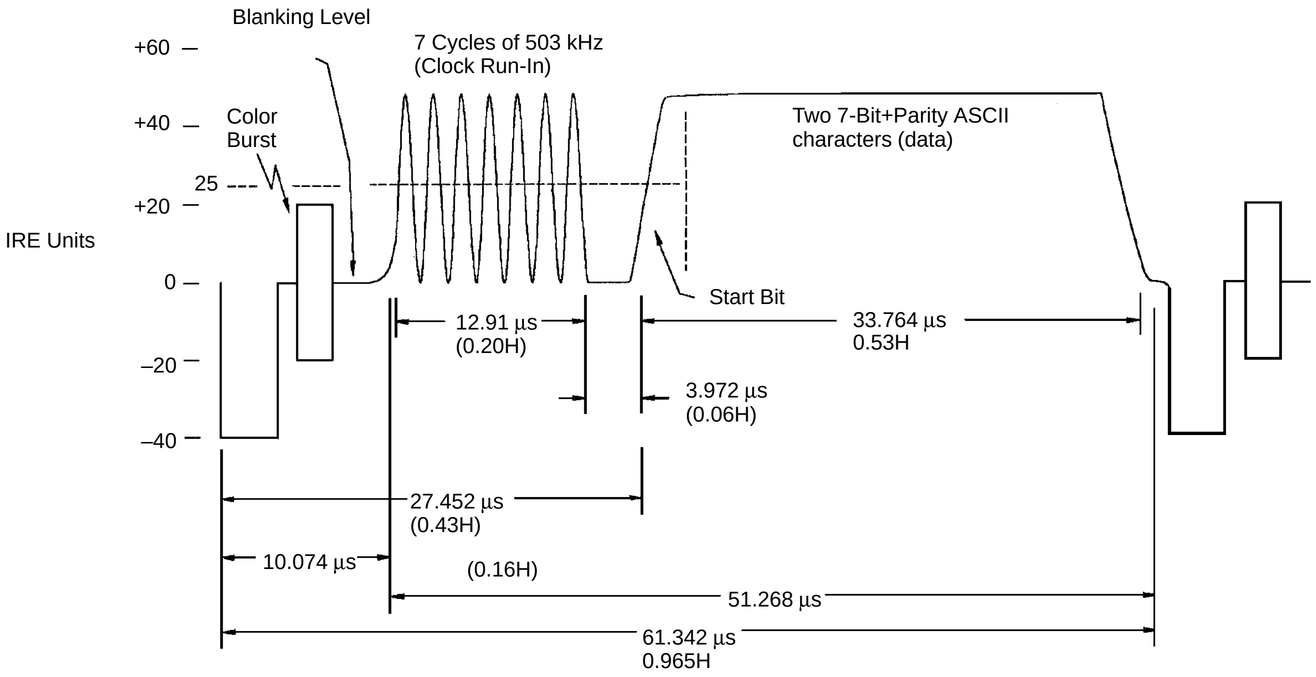


Fig. 6–1: Line 21 Field 1 Data Signal Format

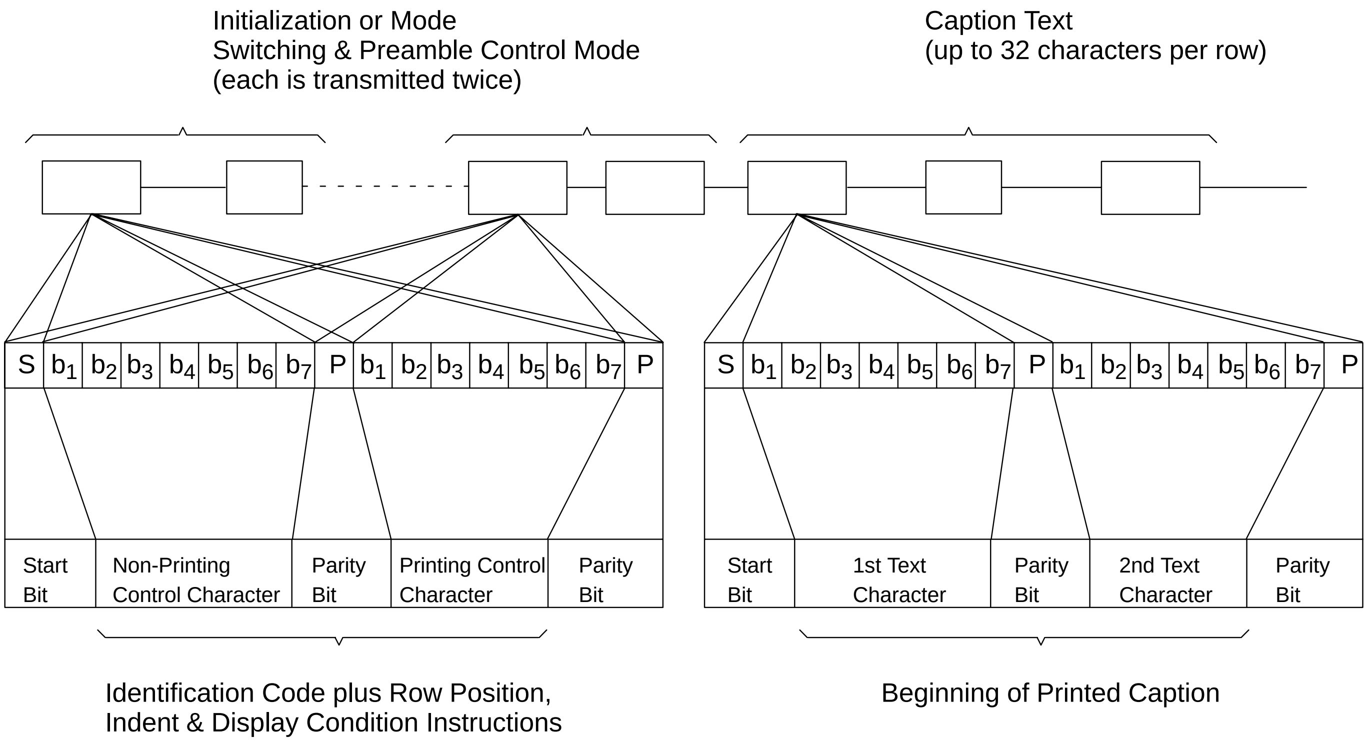

Initialization or Mode
Switching & Preamble Control Mode
(each is transmitted twice)
Caption Text
(up to 32 characters per row)
S
b1 b2 b3 b4 b5 b6 b7
P
Start Bit
Non-Printing Control Character
Parity Bit
Printing Control Character
Parity Bit
1st Text Character
2nd Text Character
Identification Code plus Row Position,
Indent & Display Condition Instructions
Beginning of Printed Caption

Fig. 6–2: Caption Row Preamble Form

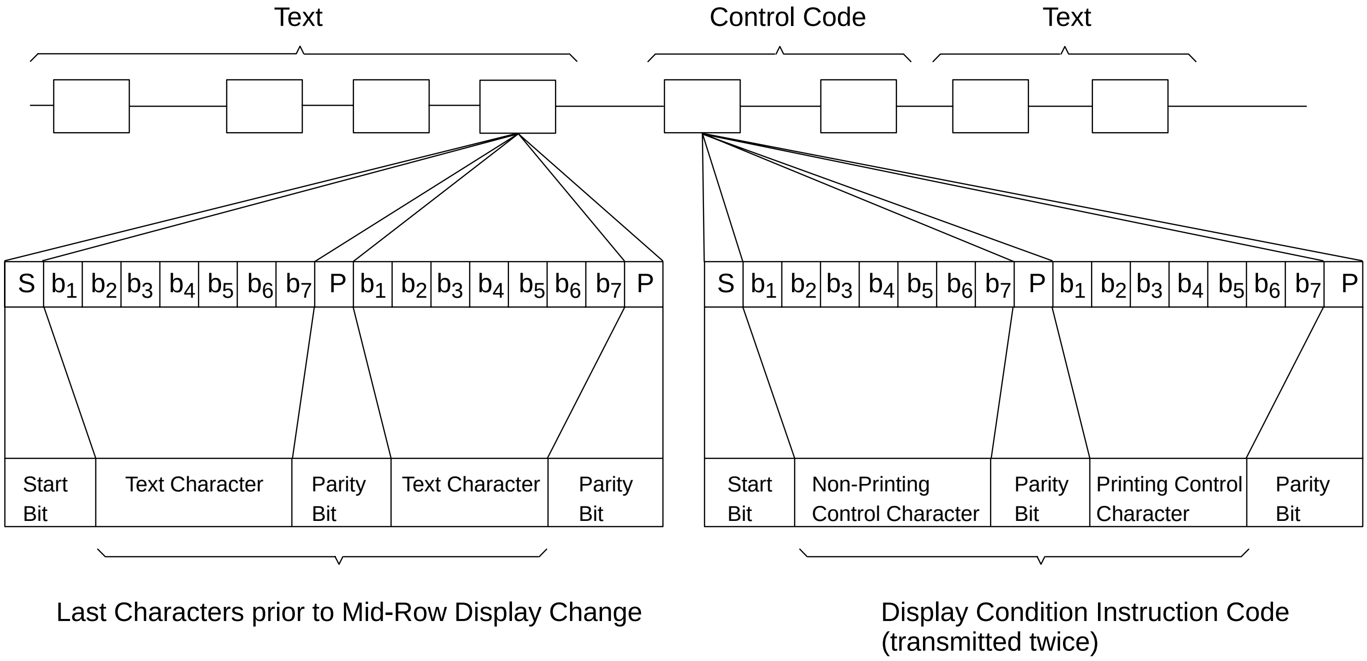

Text
Control Code
Text
S
b1 b2 b3 b4 b5 b6 b7
P
Start Bit
Text Character
Parity Bit
Non-Printing Control Character
Printing Control Character
Last Characters prior to Mid-Row Display Change
Display Condition Instruction Code
(transmitted twice)

Fig. 6–3: Mid-Caption Display – Condition Change Format

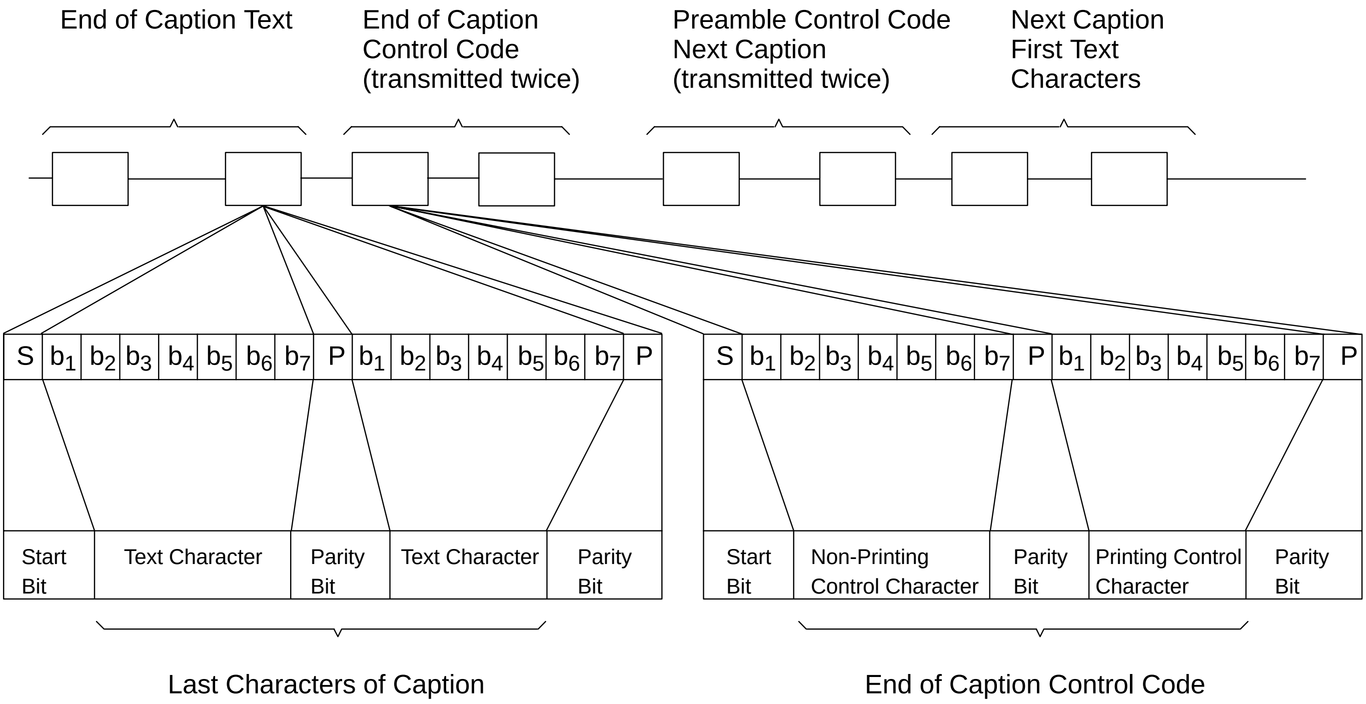

End of Caption Text
End of Caption Control Code (transmitted twice)
Preamble Control Code Next Caption (transmitted twice)
Next Caption First Text Characters
S b1 b2 b3 b4 b5 b6 b7 P b1 b2 b3 b4 b5 b6 b7 P
S b1 b2 b3 b4 b5 b6 b7 P b1 b2 b3 b4 b5 b6 b7 P
Start Bit
Text Character
Parity Bit
Text Character
Parity Bit
Start Bit
Non-Printing Control Character
Parity Bit
Printing Control Character
Parity Bit
Last Characters of Caption
End of Caption Control Code

Fig. 6–4: End of Caption and Caption Transition Format

The first character of the control code is a non-printing ASCII character (000 0000 through 001 1111) followed by a printing character (010 0000 through 111 1110). All characters that are received after a set of valid control codes are interpreted and loaded into memory as printing characters. Reception of any invalid control code will cause the system to ignore all subsequent character transmissions until it receives a valid control preamble. Character codes with bad parity result in an all-ones code being written into memory; this causes display of a box (the delete symbol) in place of the desired character which, of course, is an error.

A valid control code always begins with one of four non-printing ASCII control characters – DC1, DC2, DC3, or DC4 – followed by a printing character code which, when combined, define all the required addressing and display functions. The complete catalog of these control codes is shown on pages 61 through 62.

6.2. Closed Caption Decoder

6.2.1. Operating Modes

There are three modes of operation: Caption, Text, and TV. The TV mode of operation will disable the display of caption data, allowing the TV video (off-air, VCR, etc.) to be seen in its original form. The Text and Caption Modes black out one or more areas (called “boxes”) on the screen within which caption or text characters are displayed.

In addition, the OSD of the CCZ may be used to provide customer-specific fade-ins, such as program numbers or names, bars of analog values, etc.

6.2.2. Screen Format

The display area for box(es) and text is 15 rows high by 34 columns wide. Vertically, the box area begins on line 43 and is 195 lines high, ending at line 237 (out of 241 full active scan lines in a field). Horizontally, the box begins at 13.0 µs from the 50% point of the leading edge of sync and is 45.02 µs wide, ending at 58.01 µs. Thus, the box area conforms approximately to the standard safe title area for NTSC receivers. Text will occupy the middle 32 columns of the box, which are referred to in this document as character positions 1 through 32.

6.2.2.1. Text Mode

In Text Mode, a black box 15 rows high and 34 columns wide is constantly displayed whenever valid data (see section 6.2.7.) are being processed by the decoder, regardless of how many characters, if any, are sent, and regardless of the mode or data channel to which they are addressed. Each row of text contains a maximum of 32 characters, leaving no less than a one-column wide box on the left and right of the text.

There will never be transparent spaces or transparent rows in Text Mode. The only transparency will be around the outside of the box.

Text Mode is initiated by receipt of a Resume Text Display command. In Text Mode, the method of presenting characters depends on whether all 15 rows have been put up on the screen yet. When Text Mode has just been selected and the specified text memory is empty, the cursor starts at Row 1, Position 1 and moves down to Position 1 on the next row each time a carriage return is received until Row 15 is reached. As soon as the first 15 rows of text are on the screen, Text Mode switches to a scrolling type of presentation. With each subsequent carriage return received, the text on Row 1 is erased, text on rows 2–15 are rolled up one row, and the cursor is moved to position 1 on row 15, which will then be blank. Should new displayable characters be received during the time that rows 2–15 are rolling upwards, the new characters are seen appearing from top to bottom, as each text row is 13 dots high and rolls up at a rate of one dot per frame (thus, a smooth roll-up is completed in 0.433 seconds). If a carriage return is received while text is rolling up, smooth scroll is suspended, causing rows 2–15 to jump up to their final position so the new carriage return can be processed immediately.

Characters are always displayed immediately when received by the decoder. Once the cursor reaches the 32nd character position on any row, all subsequent characters received prior to a carriage return, Preamble Address Code, or Backspace will be displayed in that position, replacing any previous character occupying that address. In Text Mode, the cursor cannot be moved randomly around the screen. The cursor automatically moves one column to the right after reception of each character or Mid-Row code. Using any Preamble Address Code will move the cursor to the indicated indent position on the current row, ignoring row information contained in the address. Using a Backspace will move the cursor one column to the left, erasing the character or Mid-Row Code occupying that location. (A Backspace received when the cursor is in Position 1 will be ignored.) Using the “Text Restart” command will erase all characters on the screen and move the cursor to Row 1, Position 1.

If the reception of data for a row is interrupted by data for the alternate data channel or for Caption Mode, the display of text will resume from the same cursor position if a “Resume Text Display” command is received and no Preamble Address Code is given which would move the cursor.

A character remains displayed until

– scrolled off the top of the screen

– another character is addressed to the same screen location

– it is erased by a Backspace

– all text is erased simultaneously by receipt of a “Text Restart” command

– the user switches data channels (C1/C2)

– the user switches fields (F1/F2) or

– by loss of valid data (see Section 6.2.7.).

In addition, changing the Mode Select switch to TV will turn off the display of characters and box without erasing data from the decoder memory.

6.2.2.2. Caption Mode

In Caption Mode, text can appear only on rows 1 to 4 and 12 to 15 of the screen. Rows 5 to 11 are never used and will therefore always be transparent. Each caption row contains a maximum of 32 characters, and each character will always be preceded and followed by either another character or a box no less than one column wide.

The caption area will be transparent anywhere that:

1. no standard space character or other character has been addressed and no accompanying box is needed
2. a “transparent space” special character has been addressed which does not immediately precede or follow a displayed character.

There are three styles of presenting text in Caption Mode: roll-up, pop-on, and paint-on.

Character display varies significantly with the style used, but certain rules of character erasure are common to all styles. A character, once displayed, can be erased by addressing another character to the same screen location or by backspacing over the character from a subsequent location on the same row. The entire caption display will be erased simultaneously by receipt of an “Erase Displayed Memory” command, or by the user switching data channels (C1/C2) or fields (F1/F2), or by loss of valid data (see section 6.2.7.), or by commands specific to the type of presentation style used. Receipt of an “End of Caption” command will cause a displayed caption to become non-displayed (and vice versa) without being erased from memory. In addition, changing the Mode Select switch to TV will turn off the display of characters and box without erasing data from the decoder memory.

1. Roll-Up

Roll-up style captioning is initiated by using one of three Miscellaneous Control Codes to select the maximum number of rows displayed. A Preamble Control Code will determine the start row or bottom row. The start row must be greater than the number of rows to be scrolled or those rows not in the valid display area will not be displayed.

Regardless of the number of active rows selected, the cursor always remains on the start row selected and the text of the four most recently received rows is stored in memory. For convenience, those rows are identified as A, B, C, and D. The first row received when the memory is empty is Row A; the second row is Row B, the fifth is again Row A; etc. If the selected start row =X, each time a Carriage Return is received, those rows which occupy

Rows X–2, X–1 and X roll up one row, whether or not they are in the active display window. The text of the row occupying Row X–3 is erased, and that now-empty portion of memory is addressed to Row X ready to accept new text. The cursor is automatically placed at Pos. 1.

Given, for example, that 4–row Roll-Up has been selected. The first five rows of text received will display as follows:

ROW 1 (A — on Row 15)

ROW 1 (A — on Row 14)
ROW 2 (B — on Row 15)

ROW 1 (A — on Row 13)
ROW 2 (B — on Row 14)
ROW 3 (C — on Row 15)

ROW 1 (A — on Row 12)
ROW 2 (B — on Row 13)
ROW 3 (C — on Row 14)
ROW 4 (D — on Row 15)

ROW 2 (B — on Row 12)
ROW 3 (C — on Row 13)
ROW 4 (D — on Row 14)
ROW 5 (A — on Row 15)

If the number of roll-up rows selected were less than four, the display of Row 12 or of Rows 12 and 13 would be turned off, and the text of those rows would be erased from memory. Increasing or decreasing the number of roll-up rows immediately changes the size of the active display window appropriately turning on or off the display of those rows of text occupying Rows 12 and 13.

As in Text Mode presentation, the decoder attempts to provide a smooth scroll of displayed rows. Each row is 13 dots high and rolls up at a rate of one dot per frame, which means a scroll is completed in 0.433 seconds. Should new displayable characters be received during that time, they are seen appearing from top to bottom. If a Carriage Return is received during the 13-frame roll-up period, smooth scrolling is suspended, causing the rows to jump up immediately to their final position.

Characters are always displayed immediately when received by the decoder. Once the cursor reaches the 32nd character position on any row, all subsequent characters received prior to a Carriage Return, Preamble Address code, or Backspace will be displayed in that position, replacing any previous character occupying that address.

The cursor moves automatically one column to the right after each character or Mid-Row code received. Using any Preamble Address code will move the cursor to the indicated indent position on Row 15, ignoring row information contained in the address. Using a Backspace will move the cursor one column to the left, erasing the character or Mid-Row code occupying that location. (A Backspace received when the cursor is in Position 1 will be ignored).

The leading box appears when the first displayable character (not a transparent space) or Mid-Row code is received on a row, not when Preamble Address code, if any, is given. A row on which there are no displayable characters or Mid-Row codes will not display a box, even when rolled up between two rows which do display a box.

If the reception of data for a row is interrupted by data for the alternate data channel or for Text Mode, the display of caption text will resume from the same cursor position if a Roll-Up Caption command is received and no Preamble Address code is given which would move the cursor.

Characters remain displayed until scrolled above the top row of the active display window or until one of the standard caption erasure techniques is applied (see section 6.2.2.2.). Receipt of a Resume Caption Loading command (for Pop-on style) will not affect a roll-up display. Receipt of a Resume Direct Captioning command (for paint-on style) will cause all four roll-up rows to be active. Receipt of an End of Caption command will flip the roll-up captioning into non-displayed memory, also activating all four rows. Issuing a roll-up caption command will cause any pop-on or paint-on caption to be erased from displayed memory and/or non-displayed memory.

2. Pop-On

Pop-on style captioning is initiated by receipt of a Resume Caption Loading command. Subsequent data are loaded into a non-displayed memory and held there until an End of Caption command is received, at which point the non-displayed memory becomes the displayed memory and vice versa. (This process is often referred to as flipping memories). An End of Caption command forces the decoder into Pop-On Caption Mode if no Resume Caption Loading command has been received which would do so.

If no Preamble Address code is received, the cursor will begin at the same screen position where it was left after the previous caption (or at Row 1, Position 1 when the decoder is first turned on or after a screen erase caused by the user switching data channels or fields or by loss of valid data).

Preamble Address codes can be used to move the cursor around the screen in random order to place captions on rows 1 through 15. Carriage Returns have no effect on cursor location during caption loading.

The cursor moves automatically one column to the right after each character or Mid-Row code received. Using a Backspace will move the cursor one column to the left, erasing from memory the character or Mid-Row Code occupying that location. (A Backspace received when the cursor is in Position 1 will be ignored.) Once the cursor reaches the 32nd character position on any row, all subsequent characters received prior to an End of Caption, a Preamble Address code, or Backspace will replace any previous character at that location.

If data reception is interrupted during caption loading by data for the alternate caption channel or for Text Mode, caption loading will resume at the same cursor position if a Resume Caption Loading command is received and no Preamble Address code is given that would move the cursor.

Characters remain in non-displayed memory until an End of Caption command flips memories. The caption will be erased without being displayed upon receipt of an Erase Non-displayed Memory command or a Roll-Up caption command, or if the user switches data channels or fields, or upon the loss of valid data.

A pop-on caption, once displayed, remains displayed until one of the standard caption erasure techniques is applied (see section 6.2.2.2.) or until a Roll-Up caption command is received. Characters within a displayed pop-on caption can be replaced using the resume direct captioning command and paint-on style techniques.

3. Paint-On

Paint-on style captioning is initiated by receipt of a resume direct captioning command. Subsequent data are addressed immediately to displayed memory without need for an end of caption command.

If no Preamble address code is received, the cursor will begin at the same screen position where it was left after the previous caption (or at row 1, Position 1 when the decoder is first turned on or after a screen erase caused by the user switching data channels or fields or by loss of valid data).

Preamble Address codes can be used to move the cursor around the screen in random order to display captions on rows 1 through 15. Carriage Returns have no effect on cursor location during direct captioning.

The cursor moves automatically one column to the right after each character or Mid-Row code received. Using a Backspace will move the cursor one column to the left, erasing the character or Mid-Row code occupying that location. (A Backspace received when the cursor is in Position 1 will be ignored.) Once the cursor reaches the 32nd character position on any row, all subsequent characters received prior to a Preamble address code or Backspace will be displayed in that position, replacing any previous character occupying that address.

If the reception of data is interrupted during direct captioning by data for the alternate caption channel or for Text Mode, the display of caption text will resume at the same cursor position if a “Resume Direct Captioning” command is received and no Preamble Address Code is given which would move the cursor.

Characters remain displayed until one of the standard caption erasure techniques is applied (see section 6.2.2.2.) or until a “Roll-Up Caption” command is received. An “End of Caption” command leaves a paint-on caption fully intact in non-displayed memory. In other words, a paint-on style caption behaves precisely like a pop-on style caption which has been displayed.

6.2.3. Presentation Format

In analyzing the presentation of characters, it is convenient to think in terms of a non-visible cursor which marks the screen position at which the next event in a given mode and data channel will occur. The decoder remembers the cursor position for each mode even when data are received for a different address in the alternate mode or data channel.

6.2.4. Character Format

Characters are to be displayed on the screen in a dot matrix format. Each character will be displayed within a character “cell” which is the height and width of a single row and column.

Each cell is 13 dots high by 8 dots wide. Vertically, each dot is to be one pair of interlaced scan lines. Within a field, this design will make a character cell 13 scan lines high, but within a frame each cell will be 26 scan lines high. (The lines from each field are like-numbered pairs). There are no scan lines between character rows.

In 525 standards the vertical dots are said to be in dot rows, with the topmost dot row being numbered dot row 0 (zero) and the bottommost being dot row 12. Dot rows 0 and 12 will always be blank, providing vertical spacing between characters. An underline, if used (see section 6.2.5.), will be shown on dot row 11. Dot rows 1 through 11 are for the displayed character.

The horizontal dots of the cell are numbered from dot 0 (zero) on the left to dot 7 on the right. Dot 7 on every dot row will always be blank, except on dot row 11 when the underline attribute is assigned. This method provides horizontal spacing between characters while connecting an underline from cell to cell.

Table 6–1: Special characters (the values are hexadecimal numbers)

Data Channel		Character
1	2	
11 30	19 30	®
11 31	19 31	degree sign
11 32	19 32	1/2
11 33	19 33	inverse query
11 34	19 34	™ (trade mark)
11 35	19 35	cents sign
11 36	19 36	pounds sign
11 37	19 37	music note
11 38	19 38	lower-case a with grave accent
11 39	19 39	transparent space
11 3A	19 3A	lower-case e with grave accent
11 3B	19 3B	lower-case a with circumflex
11 3C	19 3C	lower-case e with circumflex
11 3D	19 3D	lower-case i with circumflex
11 3E	19 3E	lower-case o with circumflex
11 3F	19 3F	lower-case u with circumflex

Note: This is not a complete list of special characters.

6.2.5. Character Attributes

A character may have any or all of four attributes: color, italics, underline and flash. All of these attributes are set by control codes included in the received data. An attribute will remain in effect until another control code is received or until the end of the row is reached. Each row begins with a control code which sets the color and underline attributes. (White non-underlined is the default if no Preamble Address code is received before the first character on an empty row). Attributes are not affected by transparent spaces within a row.

All Mid-Row codes and the “Flash On” command are spacing attributes which appear in the display just as if a standard space (20H) had been received. Preamble Address codes are non-spacing and will not alter any attributes when used to position the cursor in the midst of a row of characters.

The color attribute has the highest priority and can only be changed by the Mid-Row code of another color.

Italics has the next highest priority. If characters with both color and italics are desired, the italics Mid-Row code must follow the color assignment. Any color Mid-Row Code will turn off italics.

If the least significant bit of a Preamble Address code or of a color or italics Mid-Row code is a 1 (high), underlining is turned on. If that bit is a 0 (low), underlining is off.

The flash attribute is transmitted as a Miscellaneous control code. The Flash On command will not alter the status of the color, italics, or underline attributes. However, any color or italics Mid-Row Code will turn off flash.

Thus, for example, if a red, italicized, underlined, flashing character is desired, the attributes must be received in the following order: a red Mid-Row or Preamble Address code, an italics Mid-Row code with underline bit, and the Flash On command. The character will then be preceded by three spaces (two if red was assigned via a Preamble Address code).

The available colors are white, green, blue, cyan, red, yellow, and magenta. In all cases, the background is black.

A character with italics will have a two-dot slant to the right over the vertical range of the character as defined above.

An underline attribute will draw an underline beneath a character in the same color as the character. The underline resides on dot row 11, one dot row below the lowest dot of a character. It covers the entire width of a column.

A flash attribute will blank the display of a character for 0.25 seconds once per second.

6.2.6. Control Codes

There are three different types of control codes used to identify the format, location, attributes, and display of characters: Preamble Address codes, Mid-Row codes, and Miscellaneous control codes.

Each control code consists of a pair of bytes which are always transmitted together in a single field of line 21 and which are normally transmitted twice in succession to help ensure correct reception of the control instructions. The first of the control code bytes is a non-printing character in the range 10H to 1FH. The second byte is always a printing character in the range 20H to 7FH. Any such control code pair received which has not been assigned a function is ignored. If the non-printing character in the pair is in the range 00H to 0FH, that character alone will be ignored and the second character will be treated normally.

If the second byte of a control code pair does not contain odd parity, the pair is ignored. The redundant transmission of the pair will be the instruction upon which the decoder acts.

If the first byte of the first transmission of a control code pair fails the parity check, then that byte is inserted into the currently active memory as a block character (7FH) followed by whatever the second byte is. Again, the redundant transmission of the pair will be the controlling instruction.

If the first transmission of a control code pair passes parity, it is acted upon immediately. If the next frame contains a perfect repeat of the same pair, the redundant code is ignored. If, however, the next frame contains a different but also valid control code pair, this pair, too, will be acted upon (and the decoder will expect a repeat of this second pair in the next frame). If the first byte of the expected redundant control code pair fails the parity check and the second byte is identical to the second byte in the immediately preceding pair, then the expected redundant code is ignored. If there are printing characters in place of the redundant code, they will be processed normally.

The first byte of every control code pair indicates the data channel (C1/C2) to which the command applies. If bit 3 of this byte is a “0” (xxxx0xxx), data channel 1 is indicated. If bit 3 is a “1” (xxxx1xxx), data channel 2 is indicated. Control codes which do not match the data channel switch position selected by the user, and all subsequent data related to that control code, are ignored by the decoder.

Table 6–2: Miscellaneous control codes (the values are hexadecimal numbers)

Data Channel		Mnemonic	Command Description
1	2		
14 20	1C 20	RCL	Resume Caption Loading
14 21	1C 21	BS	Backspace
14 22	1C 22		Reserved
14 23	1C 23		Reserved
14 24	1C 24	DER	Delete to End of Row
14 25	1C 25	RU2	Roll-Up Captions — 2 Rows
14 26	1C 26	RU3	Roll-Up Captions — 3 Rows
14 27	1C 27	RU4	Roll-Up Captions — 4 Rows
14 28	1C 28	FON	Flash On
14 29	1C 29	RDC	Resume Direct Captioning
14 2A	1C 2A	TR	Text Restart
14 2B	1C 2B	RTD	Resume Text Display
14 2C	1C 2C	EDM	Erase Displayed Memory
14 2D	1C 2D	CR	Carriage Return
14 2E	1C 2E	ENM	Erase Non-Displayed Memory
14 2F	1C 2F	EOC	End of Caption (Flip Memories)
17 21	1F 21	TO1	Top Offset 1 Column
17 22	1F 22	TO2	Top Offset 2 Columns
17 23	1F 23	TO3	Top Offset 3 Columns

Table 6–3: Mid-Row codes

Data Channel		Attribute Description
1	**2**	
11 20	19 20	White
11 21	19 21	White Underline
11 22	19 22	Green
11 23	19 23	Green Underline
11 24	19 24	Blue
11 25	19 25	Blue Underline
11 26	19 26	Cyan
11 27	19 27	Cyan Underline
11 28	19 28	Red
11 29	19 29	Red Underline
11 2A	19 2A	Yellow
11 2B	19 2B	Yellow Underline
11 2C	19 2C	Magenta
11 2D	19 2D	Magenta Underline
11 2E	19 2E	Italics
11 2F	19 2F	Italics Underline

Table 6–4: Standard telecaption character set

MSB LSB	2	3	4	5	6	7
0		0	@	P	ú	p
1	!	1	A	Q	a	q
2	”	2	B	R	b	r
3	#	3	C	S	c	s
4	$	4	D	T	d	t
5	%	5	E	U	e	u
6	&	6	F	V	f	v
7	’	7	G	W	g	w
8	(	8	H	X	h	x
9	)	9	I	Y	i	y
A	á	:	J	Z	j	z
B	+	;	K	[	k	ç
C	’	<	L	é	l	÷
D	–	=	M	]	m	Ñ
E	.	>	N	í	n	ñ
F	/	?	O	ó	o	█

Table 6–5: Preamble address codes (the values are hexadecimal numbers)

Row No.								
First byte of code pair	1	2	3	4	12	13	14	15
Data Channel 1	11	11	12	12	13	13	14	14
Data Channel 2	19	19	1A	1A	1B	1B	1C	1C
Second byte of code pair								
White	40	60	40	60	40	60	40	60
White Underline	41	61	41	61	41	61	41	61
Green	42	62	42	62	42	62	42	62
Green Underline	43	63	43	63	43	63	43	63
Blue	44	64	44	64	44	64	44	64
Blue Underline	45	65	45	65	45	65	45	65
Cyan	46	66	46	66	46	66	46	66
Cyan Underline	47	67	47	67	47	67	47	67
Red	48	68	48	68	48	68	48	68
Red Underline	49	69	49	69	49	69	49	69
Yellow	4A	6A	4A	6A	4A	6A	4A	6A
Yellow Underline	4B	6B	4B	6B	4B	6B	4B	6B
Magenta	4C	6C	4C	6C	4C	6C	4C	6C
Magenta Underline	4D	6D	4D	6D	4D	6D	4D	6D
White Italics	4E	6E	4E	6E	4E	6E	4E	6E
White Italics Underline	4F	6F	4F	6F	4F	6F	4F	6F
Indent 0	50	70	50	70	50	70	50	70
Indent 0 Underline	51	71	51	71	51	71	51	71
Indent 4	52	72	52	72	52	72	52	72
Indent 4 Underline	53	73	53	73	53	73	53	73
Indent 8	54	74	54	74	54	74	54	74
Indent 8 Underline	55	75	55	75	55	75	55	75
Indent 12	56	76	56	76	56	76	56	76
Indent 12 Underline	57	77	57	77	57	77	57	77
Indent 16	58	78	58	78	58	78	58	78
Indent 16 Underline	59	79	59	79	59	79	59	79
Indent 20	5A	7A	5A	7A	5A	7A	5A	7A
Indent 20 Underline	5B	7B	5B	7B	5B	7B	5B	7B
Indent 24	5C	7C	5C	7C	5C	7C	5C	7C
Indent 24 Underline	5D	7D	5D	7D	5D	7D	5D	7D
Indent 28	5E	7E	5E	7E	5E	7E	5E	7E
Indent 28 Underline	5F	7F	5F	7F	5F	7F	5F	7F

Note: All indent codes (second byte equals 50H–5FH, 70H–7FH) assign white as the color attribute.

6.2.7. Data Rejection

The decoder tends to reject data for three reasons: the data are invalid, they are signalled for the data channel not selected by the user, or they are in the line 21 field not selected by the user. Invalid data are data which fail to pass a check for odd parity or data which, having passed the parity check, are assigned no function. The effect of invalid data in control codes is covered in section 6.2.6. Other data which fail parity are always displayed in the current mode (i.e. the same mode as the preceding non-null byte) as block characters (7FH). A parity error in 45 successive frames will disable the video display and erase all memories (see section 6.2.8.). Data rejected for any other reason are ignored and, therefore, lost.

6.2.8. Automatic Display Enable/Disable

The decoder will automatically enable and disable the display of box and text in response to the presence or absence of valid data on line 21.

6.2.8.1. Enable Logic and Timing

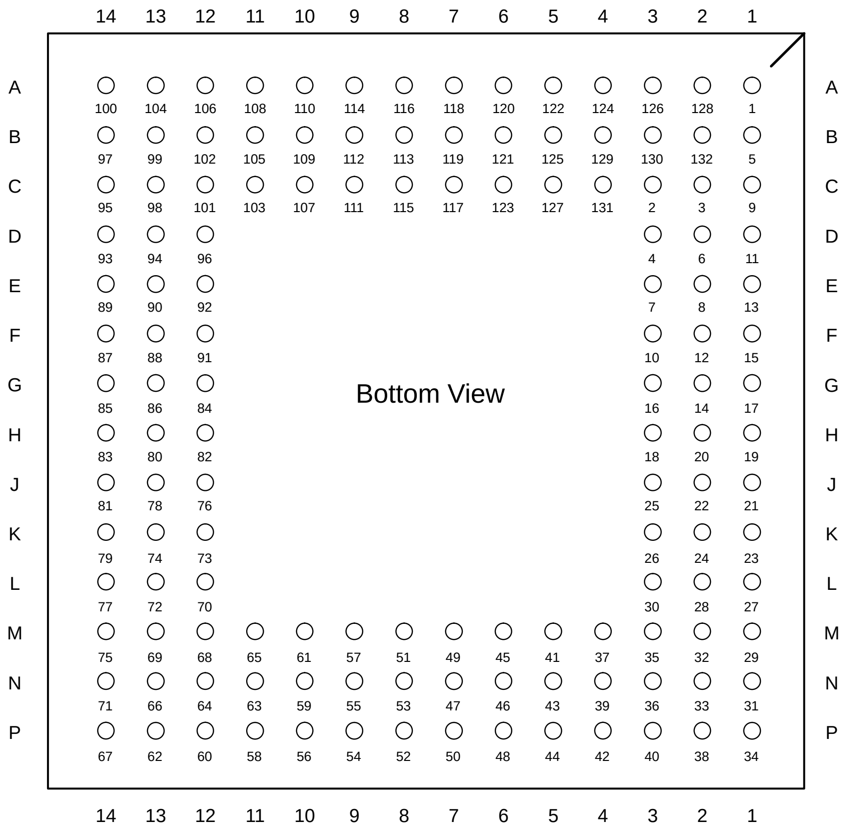
The minimum time for the display to be enabled while disabled is 15 frames. When the decoder is first turned on, or after the display has been disabled, it will not be enabled until 15 consecutive frames have been received in which both data bytes pass the check for odd parity. If, while accumulating the 15 frames, a frame occurs in which no data is detected or in which one or more data bytes have even parity, the count of consecutive frames will be reset to zero. The data contained in the 15 frames counted to enable the display are preserved and will be displayed appropriately immediately after the 15th frame.

6.2.8.2. Disable Logic and Timing

The minimum time for the display to be disabled while enabled is 45 frames.

After the display has been enabled, it will not be disabled until 45 consecutive frames have been received in which no data are detected or in which at least one data byte has even parity. If, while accumulating the 45 frames, a frame occurs in which both data bytes have odd parity, the count of consecutive frames will be reset to zero.

7. Appendix B: Pin Configuration of CPGA Package



	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
A	100	104	106	108	110	114	116	118	120	122	124	126	128	1	A
B	97	99	102	105	109	112	113	119	121	125	129	130	132	5	B
C	95	98	101	103	107	111	115	117	123	127	131	2	3	9	C
D	93	94	96									4	6	11	D
E	89	90	92									7	8	13	E
F	87	88	91									10	12	15	F
G	85	86	84									16	14	17	G
H	83	80	82									18	20	19	H
J	81	78	76									25	22	21	J
K	79	74	73									26	24	23	K
L	77	72	70									30	28	27	L
M	75	69	68	65	61	57	51	49	45	41	37	35	32	29	M
N	71	66	64	63	59	55	53	47	46	43	39	36	33	31	N
P	67	62	60	58	56	54	52	50	48	44	42	40	38	34	P
	14	13	12	11	10	9	8	7	6	5	4	3	2	1	

Fig. 7–1: Pinning of EMU CCZ 3005K in 132-pin Ceramic Package Grid Array (CPGA132F**)**

7.1. Pin Connections in CPGA132F Package

Bond	Pin	Symbol	Bond	Pin	Symbol	Bond	Pin	Symbol	Bond	Pin	Symbol
1	A–1	nc	34	P–1	nc	67	P–14	nc	100	A–14	nc
2	C–3	nc	35	M–3	P13 ↕	68	M–12	P26 ↕	101	C–12	P02 ↕
3	C–2	GNDA ←	36	N–3	nc	69	M–13	nc	102	B–12	nc
4	D–3	V_{SUPA} ←	37	M–4	P14 ↕	70	L–12	P27 ↕	103	C–11	P03 ↕
5	B–1	Half-Video →	38	P–2	nc	71	N–14	nc	104	A–13	nc
6	D–2	B_{OUT} →	39	N–4	$\overline{RESET}$ ↔	72	L–13	P30 ↕	105	B–11	P04/V_{OSD} ↕
7	E–3	nc	40	P–3	nc	73	K–12	nc	106	A–12	nc
8	E–2	G_{OUT} →	41	M–5	XTAL1 ←	74	K–13	P31 ↕	107	C–10	P05/H_{OSD} ↕
9	C–1	nc	42	P–4	nc	75	M–14	nc	108	A–11	nc
10	F–3	R_{OUT} →	43	N–5	XTAL2 ←	76	J–12	P32 ↕	109	B–10	P06 ↕
11	D–1	Fast Blank →	44	P–5	nc	77	L–14	nc	110	A–10	nc
12	F–2	H_{SYNC1} ←	45	M–6	GND ←	78	J–13	P33 ↕	111	C–9	P07 ↕
13	E–1	V_{SYNC} ←	46	N–6	nc	79	K–14	A0 →	112	B–9	D0 ↔
14	G–2	EMU ←	47	N–7	V_{SUP} ←	80	H–13	A1 →	113	B–8	D1 ↔
15	F–1	$R/\overline{W}$ →	48	P–6	TEST ←	81	J–14	A2 →	114	A–9	D2 ↔
16	G–3	$\overline{CE}$ →	49	M–7	A8 →	82	H–12	A3 →	115	C–8	D3 ↔
17	G–1	$\Phi 2$ →	50	P–7	A9 →	83	H–14	A4 →	116	A–8	D4 ↔
18	H–3	NMI →	51	M–8	A10 →	84	G–12	A5 →	117	C–7	D5 ↔
19	H–1	$\Phi 2_{CPU}$ →	52	P–8	A11 →	85	G–14	A6 →	118	A–7	D6 ↔
20	H–2	DMA →	53	N–8	A12 →	86	G–13	A7 →	119	B–7	D7 ↔
21	J–1	$\overline{NMI}_{CPU}$ ←	54	P–9	A13 →	87	F–14	IRQ →	120	A–6	nc
22	J–2	$\Phi 2_{OUT}$ →	55	N–9	A14 →	88	F–13	P34 ↕	121	B–6	nc
23	K–1	$RESET_{CPU}$ ←	56	P–10	A15 →	89	E–14	nc	122	A–5	$VIDEO_{IN}$ ←
24	K–2	$SYNC_{CPU}$ →	57	M–9	P20 ↕	90	E–13	P35 ↕	123	C–6	Slicer Cap. ←
25	J–3	CCLINE →	58	P–11	P21 ↕	91	F–12	nc	124	A–4	GND (PORQTEST) ←
26	K–3	SYNCTIP CLAMP GATE →	59	N–10	P22 ↕	92	E–12	P36 ↕	125	B–5	ADC_0 ←
27	L–1	$\overline{RUN\text{-}IN\ KEY}$ →	60	P–12	nc	93	D–14	nc	126	A–3	ADC_1 ←
28	L–2	P10 ↕	61	M–10	P23 ↕	94	D–13	P37/PWM6 ↕	127	C–5	ADC_2 ←
29	M–1	nc	62	P–13	nc	95	C–14	nc	128	A–2	nc
30	L–3	P11 ↕	63	N–11	P24 ↕	96	D–12	P00 ↕	129	B–4	ADC_3 ←
31	N–1	nc	64	N–12	nc	97	B–14	nc	130	B–3	nc
32	M–2	P12 ↕	65	M–11	P25 ↕	98	C–13	P01 ↕	131	C–4	ADC_4 ←
33	N–2	nc	66	N–13	nc	99	B–13	nc	132	B–2	nc

□	standard pin	←	input
▨	additional EMU pin	→	output
■	nc	↔	bidirectional
		↕	direction programmable

8. Data Sheet History

1. Preliminary data sheet: “CCZ 3005K Central Control Unit”, June 28, 2000, 6251-471-1PD. First release of the preliminary data sheet.

Micronas GmbH
Hans-Bunte-Strasse 19
D-79108 Freiburg (Germany)
P.O. Box 840
D-79008 Freiburg (Germany)
Tel. +49-761-517-0
Fax +49-761-517-2174
E-mail: docservice@micronas.com
Internet: www.micronas.com

Printed in Germany
Order No. 6251-471-1PD

All information and data contained in this data sheet are without any commitment, are not to be considered as an offer for conclusion of a contract, nor shall they be construed as to create any liability. Any new issue of this data sheet invalidates previous issues. Product availability and delivery are exclusively subject to our respective order confirmation form; the same applies to orders based on development samples delivered. By this publication, Micronas GmbH does not assume responsibility for patent infringements or other rights of third parties which may result from its use.
Further, Micronas GmbH reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.