

FALL 2024

RCA 2020

I2C drivers for the RCA 2020

Implementation and Test Document

RCA2020

An RCA CDP1802 COSMAC revival project

Designed by Richard van Harderwijk

I2C drivers Implementation and Test Document

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Description/Changes** |
| 1.0 | Nov 2024 | Richard van Harderwijk | First Version |
|  |  |  |  |
|  |  |  |  |

Content

[Introduction 4](#_Toc183338715)

[Implementation 5](#_Toc183338716)

[A brief description of the solution 5](#_Toc183338717)

[Implementation and driver calling 5](#_Toc183338718)

[Testing 6](#_Toc183338719)

[Use of the PCF8563 (real time clock/calendar) as testdevice 6](#_Toc183338720)

[Testresults 6](#_Toc183338721)

[I2C bus analysis 6](#_Toc183338722)

[Conclusions 7](#_Toc183338723)

[Annex A – References and Additional Documentation 8](#_Toc183338724)

[Annex B – Listings 9](#_Toc183338725)

[Controller Transmitter mode 9](#_Toc183338726)

[Controller Receiver mode 9](#_Toc183338727)

[Initialization 9](#_Toc183338728)

[Test Program 14](#_Toc183338729)

# Introduction

**About this document**

This document describes the implementation and testing of the drivers for the RCA2020 I2C interface [1].

The RCA2020 system is an SBC (Single Board Computer) based on the 1970’s RCA CDP1802 COSMAC CPU. While specific for this system, the interface should be easily portable to any other 1802 system, both the hardware and the software.

**Terminology**

The industry moved away from the old terms master and slave to *controller* and *peripheral*. These terms are used in this document, but in references you may still find the older terms.

# Implementation

## A brief description of the solution

A part of the RCA2020 system [1] is an **interface from the 1802 CPU to the I2C bus**.

The RCA2020 I2C system consists of two hardware parts:

* The core: a controller chip (PCA9564) [3], [4] connected to the CPU, as described in the design, build and test manual [1]. This interfaces the CPU to the I2C bus.
* An on-board peripheral: a date/time and alarm chip (PCF8563) [1] connected via the I2C bus.

The four registers of the controller chip are mapped to INP/OUT 4..7 (not memory mapped).

The I2C interface design criteria are described in the design document [6]

## Implementation and driver calling

Example flowcharts are excellently described in the application note of the bus controller (AN10148) [4] [6]. Three drivers are implemented:

1. I2C Controller Transmitter mode (driver code start at $7000)
2. I2C Controller Receiver mode (driver code start at $7100)
3. I2C Controller Initialization Sequence (driver code start at $7200)

The drivers are page relocatable.

The listings are in the annexes and on github [6] including the binaries, etc.

The routines expect a defined stack R(X) for the INP/OUT commands.The use of register R(7) and R(8) for calling and return codes are described in the design document, and in the listings. The routines return with SEP 5 (SCRT return).

As a side note: asking ChatGPT to create these drivers was a hallucinating experience with mnemonics and op-codes from all CPU’s except the 1802… We’re still not obsolete...

# Testing

## Use of the PCF8563 (real time clock/calendar) as testdevice

The onboard RTC chip is used as testdevice. The testprogram initializes the RTC [7], [8] just before new years eve and reads the date/time after a delay. Initialization is done by sending a 17 byte string, denoting the startregister to write (0x00) and then 16 bytes for the 16 registers of the PCF8563.

## Testresults

Find below the run and results of the testprogram. The RCA2020 uses UT4 [9], [1] as operating system-ish. The I2C routines return values are: 0x0000 indicating no errors.

......\*$P

......

......Start I2C test PCF8563 Real time clock

......TX return value: 0000

......Delay

......RX return value: 0000

......End test. Press return.

......

......\*?M F0 30

......00F0 6E2E 0D0A 0000 0000 5859 2331 0012 2380;

......0100 8080 8000 03FF 0000 0400 0001 0101 2480;

......0110 8080 8004 07FF 0000 22C2 00E0 4484 4100

......\*

Happy New year!

The yellow marked string indicates the date/time set =>

23:59:58 on Sunday Dec 31, 2023

The green marked string indicates the date/time read after the delay=>

00:00:04 on Monday Jan 1, 2024

## I2C bus analysis

A screengrab of the connected logic analyzer while transmitting is seen below. It shows the sending of the peripheral address (0x51); 0x00 to indicate start writing at PCF8563 register 0; 0x00 for regs 0, 1; 0x58 for the seconds (reg 2) and 0x59 for the minutes (reg 3).

It also indicates that the 1802 and the routines are not fast enough to saturate the I2C channel.

Afbeelding met schermopname, lijn, tekst, nummer

Automatisch gegenereerde beschrijving

## Conclusions

* The drivers work.
* The hardware (I2C part of the RCA2020) works.
* The hardware I2C controller (PCA9564) and drivers give the 1802 a robust I2C solution, easily implementable on other 1802 systems (‘only’ needing INP/OUT 4..7).
* The RTC chip works, giving the RCA2020 a real-time clock, calendar, and timers (with interrupt capability).
* The implementation of the drivers (with all error trapping) makes the drivers too slow to saturate the I2C channel. This is not a big problem since the interface is not meant for block transfer of large amounts of data.

# Annex A – References and Additional Documentation

**References**

[1] RCA2020 system/github

<https://github.com/richardvanharderwijk/1802-SBC-the-RCA2020>

[2] MPM-201 User Manual for the CDP1802 COSMAC Microprocessor <http://www.bitsavers.org/pdf/rca/1802/MPM-201A_CDP1802_User_Manual_1976.pdf>

[3] Data sheet PCA9564 (I2C-bus controller)

<https://www.nxp.com/docs/en/data-sheet/PCA9564.pdf>

[4] AN10148: PCA9564 – I2C-bus controller <https://www.nxp.com/docs/en/application-note/AN10148.pdf>

[5] The Book Of I2C – Randall Hyde; ISBN 9781718502468

[6] RCA2020 I2C/github (the design document, listing, code, etc)

<https://github.com/richardvanharderwijk/1802-I2C-interface-for-RCA2020>

[7] Data sheet PCF8563 (date/time and alarm chip)

<https://www.nxp.com/docs/en/data-sheet/PCF8563.pdf>

[8] UM10301: User Manual for NXP Real Time Clocks PCF8563, …

<https://www.nxp.com/docs/en/user-guide/UM10301.pdf>

[9] EVALUATION KIT MANUAL FOR THE RCA CDP1802 COSMAC MICROPROCESSOR MPM-203

<http://www.bitsavers.org/components/rca/cosmac/MPM-203_CDP1802_Evaluation_Kit_Manual_Sep76.pdf>

**Additional Documentation**

Libraries for using various I2C devices with an 1802-Mini with the PIO and I2C expansion boards. – Various contributors from the COSMAC ELF Group [cosmacelf@groups.io](mailto:cosmacelf@groups.io) - <https://github.com/fourstix/Elfos-I2C-Libraries/tree/main?tab=readme-ov-file>

# Annex B – Listings

## Controller Transmitter mode

## Controller Receiver mode

## Initialization

0000 ;

0000 ;

0000 ; Driver: I2C Controller Transmitter mode

0000 ; For 1802 CPU - RCA 2020 SBC with PCA9564 Controller

0000 ; Version: 0.1

0000 ; Date: Oct 31, 2024

0000 ; Author: Richard van Harderwijk

0000 ;

0000 ; PCA9564 connected via INP/OUT 4..7

0000 ; Doc: https://github.com/richardvanharderwijk/1802-I2C-interface-for-RCA2020

0000 ; Doc: AN10148 application note, figure 6a

0000 ;

0000 ; Pre: R(8) is pointer to string of bytes to send; R(7).1 is peripheral address; R(7).0 is #bytes to send

0000 ; Post on success: R(7).1 and R(7).0 contain 0x00

0000 ; Post on error: R(7).1 is Expected I2CSTA value; R(7).0 is Actual I2CSTA returned value

0000 ; End with SCRT return D5

0000 ;

0000 ;

7000 .ORG $7000

7000 BEGIN\_TX:

7000 87 GLO 7 ; test on #bytes to send > 0

7001 3A 09 BNZ start\_tx

7003 F8 01 LDI 0x01

7005 A7 PLO 7

7006 B7 PHI 7

7007 30 74 BR end\_tx ; exit w/ error

7009 START\_TX:

7009 F8 E4 LDI 0xE4 ; generate I2C start command

700B 73 STXD

700C 73 STXD

700D 60 IRX

700E 67 OUT 7 ; I2CCON register

700F CHK\_SI\_1: ; wait for SI = 1

700F 6F INP 7

7010 FA 08 ANI 0x08

7012 32 0F BZ chk\_SI\_1

7014 6C INP 4 ; I2CSTA register, check

7015 FB 08 XRI 0x08

7017 32 20 BZ continue1 ; ok, continue

7019 F0 LDX

701A A7 PLO 7

701B F8 08 LDI 0x08

701D B7 PHI 7

701E 30 74 BR end\_tx ; exit w/ error

7020 CONTINUE1:

7020 97 GHI 7 ; peripheral address to I2CDAT

7021 FE SHL ; Shift Left, address in MSB bits [7..1]

7022 FA FE ANI 0xFE ; R/W bit = 0 [LSB]

7024 73 STXD

7025 73 STXD

7026 60 IRX

7027 65 OUT 5 ; I2CDAT register

7028 F8 C4 LDI 0xC4 ; I2C send peripheral address + write command

702A 73 STXD

702B 73 STXD

702C 60 IRX

702D 67 OUT 7

702E CHK\_SI\_2: ; wait for SI=1

702E 6F INP 7

702F FA 08 ANI 0x08

7031 32 2E BZ chk\_SI\_2

7033 6C INP 4 ; I2CSTA register, check

7034 FB 18 XRI 0x18

7036 32 3F BZ continue2

7038 F0 LDX

7039 A7 PLO 7

703A F8 18 LDI 0x18

703C B7 PHI 7

703D 30 74 BR end\_tx ; exit w/ error

703F CONTINUE2: ; loop for byte string to send

703F 48 LDA 8 ; load data to I2CDAT

7040 73 STXD

7041 73 STXD

7042 60 IRX

7043 65 OUT 5

7044 F8 C4 LDI 0xC4 ; send data command

7046 73 STXD

7047 73 STXD

7048 60 IRX

7049 67 OUT 7

704A CHK\_SI\_3: ; wait for SI = 1, poll for transmission finished

704A 6F INP 7

704B FA 08 ANI 0x08

704D 32 4A BZ chk\_SI\_3

704F 6C INP 4 ; I2CSTA register, check

7050 FB 28 XRI 0x28

7052 32 5B BZ continue3

7054 F0 LDX

7055 A7 PLO 7

7056 F8 28 LDI 0x28

7058 B7 PHI 7

7059 30 74 BR end\_tx ; exit w/ error

705B CONTINUE3:

705B 27 DEC 7 ; more bytes to send?

705C 87 GLO 7

705D 3A 3F BNZ continue2

705F F8 D4 LDI 0xD4 ; generate stop command

7061 73 STXD

7062 73 STXD

7063 60 IRX

7064 67 OUT 7

7065 6F INP 7 ; I2CCON register, check STO bit = 0

7066 FA 10 ANI 0x10

7068 32 70 BZ continue4

706A F8 02 LDI 0x02

706C A7 PLO 7

706D B7 PHI 7

706E 30 74 BR end\_tx ; exit w/ error

7070 CONTINUE4:

7070 F8 00 LDI 0x00 ; exit w/ success :-)

7072 A7 PLO 7

7073 B7 PHI 7

7074 END\_TX:

7074 D5 SEP 5

7075 ;

7075 ;

7075 ; Driver: I2C Controller Receiver mode

7075 ; For 1802 CPU - RCA 2020 SBC with PCA9564 Controller

7075 ; Version: 0.1

7075 ; Date: Oct 31, 2024

7075 ; Author: Richard van Harderwijk

7075 ;

7075 ; PCA9564 connected via INP/OUT 4..7

7075 ; Doc: https://github.com/richardvanharderwijk/1802-I2C-interface-for-RCA2020

7075 ; Doc: AN10148 application note, figure 7a

7075 ;

7075 ; Pre: R(8) is pointer to memory allocated to store bytes received;

7075 ; Pre: R(7).1 is peripheral address; R(7).0 is #bytes to receive

7075 ; Post on success: R(7).1 and R(7).0 contain 0x00

7075 ; Post on error: R(7).1 is Expected I2CSTA value; R(7).0 is Actual I2CSTA returned value

7075 ; End with SCRT return D5

7075 ;

7075 ;

7100 .ORG $7100

7100 BEGIN\_RX:

7100 87 GLO 7 ; test on #bytes to receive > 0

7101 3A 09 BNZ start\_rx

7103 F8 01 LDI 0x01

7105 A7 PLO 7

7106 B7 PHI 7

7107 30 90 BR end\_rx ; exit w/ error

7109 START\_RX:

7109 F8 E4 LDI 0xE4 ; generate I2C start command

710B 73 STXD

710C 73 STXD

710D 60 IRX

710E 67 OUT 7 ; I2CCON register

710F CHK\_SI\_4: ; wait for SI = 1

710F 6F INP 7

7110 FA 08 ANI 0x08

7112 32 0F BZ chk\_SI\_4

7114 6C INP 4 ; I2CSTA register, check

7115 FB 08 XRI 0x08

7117 32 20 BZ continue5 ; ok, continue

7119 F0 LDX

711A A7 PLO 7

711B F8 08 LDI 0x08

711D B7 PHI 7

711E 30 90 BR end\_rx ; exit w/ error

7120 CONTINUE5:

7120 97 GHI 7 ; peripheral address to I2CDAT

7121 FE SHL ; Shift Left, address in MSB bits [7..1]

7122 F9 01 ORI 0x01 ; R/W bit = 1 [LSB]

7124 73 STXD

7125 73 STXD

7126 60 IRX

7127 65 OUT 5 ; I2CDAT register

7128 F8 C4 LDI 0xC4 ; I2C send peripheral address + write command

712A 73 STXD

712B 73 STXD

712C 60 IRX

712D 67 OUT 7

712E CHK\_SI\_5: ; wait for SI=1

712E 6F INP 7

712F FA 08 ANI 0x08

7131 32 2E BZ chk\_SI\_5

7133 6C INP 4 ; I2CSTA register, check

7134 FB 40 XRI 0x40

7136 32 3F BZ continue6

7138 F0 LDX

7139 A7 PLO 7

713A F8 40 LDI 0x40

713C B7 PHI 7

713D 30 90 BR end\_rx ; exit w/ error

713F CONTINUE6:

713F 27 DEC 7 ; last byte of receive is not acknowledged

7140 87 GLO 7 ; if only one byte to read, skip read loop

7141 32 61 BZ read\_last\_byte

7143 CONTINUE7: ; loop read all but last byte

7143 F8 C4 LDI 0xC4 ; read data command

7145 73 STXD

7146 73 STXD

7147 60 IRX

7148 67 OUT 7

7149 CHK\_SI\_6: ; wait for SI = 1, poll for transmission finished

7149 6F INP 7

714A FA 08 ANI 0x08

714C 32 49 BZ chk\_SI\_6

714E 6C INP 4 ; I2CSTA register, check

714F FB 50 XRI 0x50

7151 32 5A BZ continue8

7153 F0 LDX

7154 A7 PLO 7

7155 F8 50 LDI 0x50

7157 B7 PHI 7

7158 30 90 BR end\_rx ; exit w/ error

715A CONTINUE8:

715A 6D INP 5 ; read data from I2C bus from I2CDAT

715B 58 STR 8 ; store received I2C data via R(8) pointer

715C 18 INC 8

715D 27 DEC 7 ; more bytes to receive?

715E 87 GLO 7

715F 3A 43 BNZ continue7

7161 READ\_LAST\_BYTE:

7161 F8 44 LDI 0x44 ; read data command

7163 73 STXD ; to read last byte, not acknowledged

7164 73 STXD

7165 60 IRX

7166 67 OUT 7

7167 CHK\_SI\_7: ; wait for SI = 1, poll for transmission finished

7167 6F INP 7

7168 FA 08 ANI 0x08

716A 32 67 BZ chk\_SI\_7

716C 6C INP 4 ; I2CSTA register, check

716D FB 58 XRI 0x58

716F 32 78 BZ continue9

7171 F0 LDX

7172 A7 PLO 7

7173 F8 58 LDI 0x58

7175 B7 PHI 7

7176 30 90 BR end\_rx ; exit w/ error

7178 CONTINUE9:

7178 6D INP 5 ; read last byte from I2C bus from I2CDAT

7179 58 STR 8 ; store received I2C data via R(8) pointer

717A 18 INC 8

717B F8 D4 LDI 0xD4 ; generate stop command

717D 73 STXD

717E 73 STXD

717F 60 IRX

7180 67 OUT 7

7181 6F INP 7 ; I2CCON register, check STO bit

7182 FA 10 ANI 0x10

7184 32 8C BZ continue10

7186 F8 02 LDI 0x02

7188 A7 PLO 7

7189 B7 PHI 7

718A 30 90 BR end\_rx ; exit w/ error

718C CONTINUE10:

718C F8 00 LDI 0x00 ; exit w/ success :-)

718E A7 PLO 7

718F B7 PHI 7

7190 END\_RX:

7190 D5 SEP 5

7191 ;

7191 ;

7191 ; Driver: I2C Controller Initialization Sequence

7191 ; For 1802 CPU - RCA 2020 SBC with PCA9564 Controller

7191 ; Version: 0.1

7191 ; Date: Oct 31, 2024

7191 ; Author: Richard van Harderwijk

7191 ;

7191 ; PCA9564 connected via INP/OUT 4..7

7191 ; Doc: https://github.com/richardvanharderwijk/1802-I2C-interface-for-RCA2020

7191 ; Doc: AN10148 application note, figure 5

7191 ;

7191 ; Pre: R(7).1 future use (own peripheral address); R(7).0 future use (clock frequency)

7191 ; In this version clock frequency is set at 88 kHz

7191 ; Post on success: R(7).1 and R(7).0 contain 0x00

7191 ; Post on error: future use, in this version no test

7191 ; End with SCRT return D5

7191 ;

7191 ;

7200 .ORG $7200

7200 BEGIN\_INIT:

7200 F8 FF LDI 0xFF ; 0xFF in TimeOut register

7202 73 STXD

7203 73 STXD

7204 60 IRX

7205 64 OUT 4 ; I2CTO register

7206 F8 64 LDI 0x64 ; own peripheral address

7208 73 STXD

7209 73 STXD

720A 60 IRX

720B 66 OUT 6 ; I2CADR register

720C F8 44 LDI 0x44 ; Enable Serial IO, 88 kHz

720E 73 STXD

720F 73 STXD

7210 60 IRX

7211 67 OUT 7 ; I2CCON register

7212 F8 2F LDI 0x2F ; wait for oscillator startup

7214 DELAY1:

7214 FF 01 SMI 0x01

7216 3A 14 BNZ delay1

7218 F8 C4 LDI 0xC4 ; peripheral receiver mode

721A 73 STXD

721B 73 STXD

721C 60 IRX

721D 67 OUT 7 ; I2CCON register

721E F8 00 LDI 0x00 ; exit always w/ success :-)

7220 A7 PLO 7

7221 B7 PHI 7

7222 END\_INIT:

7222 D5 SEP 5

BEGIN\_TX: 7000 DEFINED AT LINE 31

START\_TX: 7009 DEFINED AT LINE 40

> USED AT LINE 34

CHK\_SI\_1: 700F DEFINED AT LINE 47

> USED AT LINE 50

CONTINUE1: 7020 DEFINED AT LINE 61

> USED AT LINE 54

CHK\_SI\_2: 702E DEFINED AT LINE 75

> USED AT LINE 78

CONTINUE2: 703F DEFINED AT LINE 89

> USED AT LINE 82

> USED AT LINE 119

CHK\_SI\_3: 704A DEFINED AT LINE 102

> USED AT LINE 105

CONTINUE3: 705B DEFINED AT LINE 116

> USED AT LINE 109

CONTINUE4: 7070 DEFINED AT LINE 135

> USED AT LINE 129

END\_TX: 7074 DEFINED AT LINE 140

> USED AT LINE 38

> USED AT LINE 59

> USED AT LINE 87

> USED AT LINE 114

> USED AT LINE 133

BEGIN\_RX: 7100 DEFINED AT LINE 169

START\_RX: 7109 DEFINED AT LINE 179

> USED AT LINE 172

CHK\_SI\_4: 710F DEFINED AT LINE 186

> USED AT LINE 189

CONTINUE5: 7120 DEFINED AT LINE 200

> USED AT LINE 193

CHK\_SI\_5: 712E DEFINED AT LINE 214

> USED AT LINE 217

CONTINUE6: 713F DEFINED AT LINE 228

> USED AT LINE 221

CONTINUE7: 7143 DEFINED AT LINE 233

> USED AT LINE 261

CHK\_SI\_6: 7149 DEFINED AT LINE 240

> USED AT LINE 243

CONTINUE8: 715A DEFINED AT LINE 254

> USED AT LINE 247

READ\_LAST\_BYTE: 7161 DEFINED AT LINE 263

> USED AT LINE 231

CHK\_SI\_7: 7167 DEFINED AT LINE 270

> USED AT LINE 273

CONTINUE9: 7178 DEFINED AT LINE 284

> USED AT LINE 277

CONTINUE10: 718C DEFINED AT LINE 303

> USED AT LINE 297

END\_RX: 7190 DEFINED AT LINE 308

> USED AT LINE 176

> USED AT LINE 198

> USED AT LINE 226

> USED AT LINE 252

> USED AT LINE 282

> USED AT LINE 301

BEGIN\_INIT: 7200 DEFINED AT LINE 336

DELAY1: 7214 DEFINED AT LINE 357

> USED AT LINE 359

END\_INIT: 7222 DEFINED AT LINE 371

## Test Program

0000 .ORG $0000

0000 ;

0000 ;

0000 ; Testprogram for I2C Driver with PCF8563 RealTime Clock

0000 ; For 1802 CPU - RCA 2020 SBC with PCA9564 Controller

0000 ; Version: 0.1

0000 ; Date: Nov 17, 2024

0000 ; Author: Richard van Harderwijk

0000 ;

0000 ; Uses UT4 and I2C driver routines

0000 ; Doc: https://github.com/richardvanharderwijk/1802-I2C-interface-for-RCA2020

0000 ;

0000 ; Uses SCRT

0000 ;

0000 ;

0000 BEGIN\_TST:

0000 ; enter here at $0000 with X, P = 0, 0 via UT4 (int disabled)

0000 INIT:

0000 F8 7F LDI 0x7F ; init stack in R2

0002 A2 PLO 2

0003 F8 FE LDI 0xFE

0005 B2 PHI 2

0006 E2 SEX 2

0007 ;

0007 F8 00 LDI 0x00 ; init PC in R3

0009 B3 PHI 3

000A F8 0E LDI 0x0E

000C A3 PLO 3

000D D3 SEP 3

000E ;

000E F8 C3 LDI 0xC3 ; init SCRT call in R4, return in R5

0010 B4 PHI 4

0011 B5 PHI 5

0012 F8 01 LDI 0x01

0014 A4 PLO 4

0015 F8 11 LDI 0x11

0017 A5 PLO 5

0018 ;

0018 RUN\_TST:

0018 F8 00 LDI 0x00 ; init pointer to text to print

001A B9 PHI 9

001B F8 82 LDI 0x82

001D A9 PLO 9

001E ;

001E PRINT1:

001E 49 LDA 9 ; get byte to print and put in R(F).1

001F BF PHI F

0020 32 27 BZ cont1 ; exit if byte = 0x00 (EOF)

0022 D4 SEP 4

0023 C2 A4 DW 0xC2A4 ; SCRT to UT4: TYPE @ 0xC2A4

0025 30 1E BR print1

0027 CONT1:

0027 ;

0027 ; initialization routine of I2C controller

0027 D4 SEP 4 ; SCRT to I2C controller init

0028 72 00 DW 0x7200

002A ; send initialization string to 8563 RTC

002A F8 00 LDI 0x00 ; init pointer to bytes to send

002C B8 PHI 8

002D F8 F5 LDI 0xF5

002F A8 PLO 8

0030 ;

0030 F8 51 LDI 0x51 ; peripheral/RTC address (0xA2 write)

0032 B7 PHI 7

0033 F8 11 LDI 0x11 ; 17 bytes string for RTC registers

0035 A7 PLO 7

0036 ;

0036 D4 SEP 4 ; SCRT to I2C transmitter routine

0037 70 00 DW 0x7000

0039 ;

0039 97 GHI 7 ; print return values

003A BF PHI F

003B D4 SEP 4

003C C2 AE DW 0xC2AE

003E 87 GLO 7

003F BF PHI F

0040 D4 SEP 4

0041 C2 AE DW 0xC2AE

0043 ; 8563 RTC initialization ready

0043 ;

0043 F8 00 LDI 0x00 ; init pointer to text to print

0045 B9 PHI 9 ; print delay message

0046 F8 BE LDI 0xBE

0048 A9 PLO 9

0049 ;

0049 PRINT2:

0049 49 LDA 9 ; get byte to print and put in R(F).1

004A BF PHI F

004B 32 52 BZ cont2 ; exit if byte = 0x00 (EOF)

004D D4 SEP 4

004E C2 A4 DW 0xC2A4 ; SCRT to UT4: TYPE @ 0xC2A4

0050 30 49 BR print2

0052 CONT2:

0052 ;

0052 F8 FF LDI 0xFF ; delay

0054 B9 PHI 9

0055 DELAY2:

0055 29 DEC 9

0056 C4 NOP

0057 C4 NOP

0058 C4 NOP

0059 C4 NOP

005A 99 GHI 9

005B 3A 55 BNZ delay2

005D ;

005D F8 51 LDI 0x51 ; peripheral/RTC address (0xA3 read)

005F B7 PHI 7

0060 F8 11 LDI 0x11 ; 17 bytes string for RTC registers to read

0062 A7 PLO 7

0063 ;

0063 ; R(8) already points after init (sent) string for read

0063 D4 SEP 4 ; SCRT to I2C receiver routine

0064 71 00 DW 0x7100

0066 ;

0066 97 GHI 7 ; print return values

0067 BF PHI F

0068 D4 SEP 4

0069 C2 AE DW 0xC2AE

006B 87 GLO 7

006C BF PHI F

006D D4 SEP 4

006E C2 AE DW 0xC2AE

0070 ; 8563 RTC read registers after delay ready

0070 ;

0070 F8 00 LDI 0x00 ; init pointer to text to print

0072 B9 PHI 9

0073 F8 D9 LDI 0xD9

0075 A9 PLO 9

0076 ;

0076 PRINT3:

0076 49 LDA 9 ; get byte to print and put in R(F).1

0077 BF PHI F

0078 32 7F BZ cont3 ; exit if byte = 0x00 (EOF)

007A D4 SEP 4

007B C2 A4 DW 0xC2A4 ; SCRT to UT4: TYPE @ 0xC2A4

007D 30 76 BR print3

007F CONT3:

007F ;

007F END\_TST:

007F C0 C1 00 LBR 0xC100 ; Jump to UT4 to examine return string in memory

0082 ;

0082 ;

0082 BEGIN\_TXT:

0082 0D 0A 53 74 61 72 74 20 49 32 43 20 74 65 73 74 20 50 43 46 38 35 36 33 20 52 65 61 6C 20 74 69 6D 65 20 63 6C 6F 63 6B 0D 0A 54 58 20 72 65 74 75 72 6E 20 76 61 6C 75 65 3A 20 00 DB 0x0D,0x0A,"Start I2C test PCF8563 Real time clock",0x0D,0x0A,"TX return value: ",0x00

00BE DELAY\_TXT:

00BE 0D 0A 44 65 6C 61 79 0D 0A 52 58 20 72 65 74 75 72 6E 20 76 61 6C 75 65 3A 20 00 DB 0x0D,0x0A,"Delay",0x0D,0x0A,"RX return value: ",0x00

00D9 END\_TXT:

00D9 0D 0A 45 6E 64 20 74 65 73 74 2E 20 50 72 65 73 73 20 72 65 74 75 72 6E 2E 0D 0A 00 DB 0x0D,0x0A,"End test. Press return.",0x0D,0x0A,0x00

00F5 INIT\_CLOCK\_STRING:

00F5 00 00 00 58 59 23 31 00 12 23 80 80 80 80 00 03 FF DB 0x00,0x00,0x00,0x58,0x59,0x23,0x31,0x00,0x12,0x23,0x80,0x80,0x80,0x80,0x00,0x03,0xFF

BEGIN\_TST: 0000 DEFINED AT LINE 20

INIT: 0000 DEFINED AT LINE 24

RUN\_TST: 0018 DEFINED AT LINE 46

PRINT1: 001E DEFINED AT LINE 52

> USED AT LINE 58

CONT1: 0027 DEFINED AT LINE 59

> USED AT LINE 55

PRINT2: 0049 DEFINED AT LINE 95

> USED AT LINE 101

CONT2: 0052 DEFINED AT LINE 102

> USED AT LINE 98

DELAY2: 0055 DEFINED AT LINE 106

> USED AT LINE 113

PRINT3: 0076 DEFINED AT LINE 140

> USED AT LINE 146

CONT3: 007F DEFINED AT LINE 147

> USED AT LINE 143

END\_TST: 007F DEFINED AT LINE 151

BEGIN\_TXT: 0082 DEFINED AT LINE 157

DELAY\_TXT: 00BE DEFINED AT LINE 160

END\_TXT: 00D9 DEFINED AT LINE 163

INIT\_CLOCK\_STRING: 00F5 DEFINED AT LINE 166