



MOTOROLA
intelligence everywhere™

digitaldna™

Freescale Semiconductor, Inc.

M68HC08 *Microcontrollers*

*Remote Keyless
Entry (RKE)
Reference Design
Using the
MC68HC908RF2*

*Designer Reference
Manual*

DRM005/D
5/2002

WWW.MOTOROLA.COM/SEMICONDUCTORS



Freescale Semiconductor, Inc.

Freescale Semiconductor, Inc.

**For More Information On This Product,
Go to: www.freescale.com**

Remote Keyless Entry (RKE) Reference Design Using the MC68HC908RF2

By: Andrea Martini
ESSEPIE S.r.l. — Studio di progettazione
Via Artigianato,5
24044 Dalmine (BG) — Italy

Email: essepie@wavenet.it
Web: <http://www.elektronikladen.de/en>



Motorola and the Stylized M Logo are registered trademarks of Motorola, Inc.
DigitalDNA is a trademark of Motorola, Inc.

© Motorola, Inc., 2002

Remote Keyless Entry (RKE) Reference Design Using the MC68HC908RF2

DRM005

MOTOROLA

**For More Information On This Product,
Go to: www.freescale.com**

3

Revision History

To provide the most up-to-date information, the revision of our documents on the World Wide Web will be the most current. Your printed copy may be an earlier revision. To verify you have the latest information available, refer to:

<http://www.motorola.com/semiconductors>

The following revision history table summarizes changes contained in this document. For your convenience, the page number designators have been linked to the appropriate location.

Revision History

Date	Revision Level	Description	Page Number(s)
May, 2002	N/A	Initial release	N/A

List of Sections

Section 1. General Description	11
Section 2. Motorola RF Components	17
Section 3. Reference Board Hardware	21
Section 4. HC908RF2 for Windows Installation	33
Section 5. Remote Programming Commands	43
Appendix A. Source Code	59
Appendix B. Schematics	135
Appendix C. Board Layouts	139
Appendix D. Bill of Materials	147



Table of Contents

Section 1. General Description

1.1	Contents	11
1.2	Introduction	11
1.3	Reference Board	12
1.4	Reference Goals	13

Section 2. Motorola RF Components

2.1	Contents	17
2.2	Introduction	17
2.3	MC68HC908RF2	17
2.4	MC33592/3	19
2.5	MC33491	19

Section 3. Reference Board Hardware

3.1	Contents	21
3.2	Introduction	21
3.3	Mother Board Description	22
3.4	Remote Description	24
3.5	Receiver Board Description	24
3.6	Transmitter	28
3.7	Mother Board Display Menu	30

Section 4. HC908RF2 for Windows Installation

4.1	Contents	33
4.2	Introduction	33
4.3	Installation Screens	33

Section 5. Remote Programming Commands

5.1	Contents	43
5.2	Introduction	43
5.3	Communication Protocol	43
5.4	Command Set Parameters	44
5.4.1	Erase Programming Algorithm Memory (EPG)	45
5.4.2	Erase TX S19 Memory (ETX)	46
5.4.3	Erase User Code Memory (EUM)	47
5.4.4	Initialize MC33592/3 Registers (IRR)	48
5.4.5	Load Programming Algorithm (LPG)	49
5.4.6	Load TX S19 File (LTX)	51
5.4.7	Product ID (PID)	53
5.4.8	Read MC33592/3 Registers (RRR)	54
5.4.9	Update User Code (UUC)	55
5.4.10	MC33592/3 Registers (WRR)	57

Appendix A. Source Code

A.1	Contents	59
A.2	Mother Board Firmware	60
A.2.1	UK509.ASM	60
A.2.2	UK509 PRM	124
A.3	Transmitter Firmware	125

Appendix B. Schematics

Appendix C. Board Layouts

Appendix D. Bill of Materials

D.1	Contents	147
D.2	Introduction	147
D.3	Transmitter	148
D.4	Receiver	149
D.5	Mother Board	150

Designer Reference Manual — Remote Keyless Entry

List of Figures and Tables

Figure	Title	Page
1-1	Transmitter	13
1-2	Receiver Board	15
1-3	Mother Board	16
2-1	MC33492 Pin Assignments	20
2-2	14-Lead TSSOP Pictorial	20
3-1	Mother Board Block Diagram	23
3-2	Remote Block Diagram	24
3-3	Receiver Board Block Diagram	25
3-4	Receiver Board	25
3-5	Receiver Board Schematic	26
3-6	Transmitter	28
3-7	Transmitter Schematic	29
3-8	Mother Board Display Menu	31
4-1	Startup Screen	34
4-2	Welcome Screen	35
4-3	License Agreement	36
4-4	Choose Destination Location Screen	37
4-5	Setup Type	38
4-6	Select Program Folder	39
4-7	Install Files	40
4-8	Finish	41
5-1	Command Packet Format	44
5-2	Command Packet Format	44



List of Figures and Tables

Figure	Title	Page
B-1	Mother Board Schematic	136
B-2	Receiver Schematic	137
B-3	Transmitter Schematic	138
C-1	Transmitter Layout (Top View)	140
C-2	Transmitter Layout (Bottom View)	141
C-3	Receiver Layout (Top View)	142
C-4	Receiver Layout (Bottom View)	143
C-5	Mother Board Layout (Top View)	144
C-6	Mother Board Layout (Bottom View).	145
Table	Title	Page
2-1	MC33492 Ordering Information	20

Section 1. General Description

1.1 Contents

1.2	Introduction	11
1.3	Reference Board.	12
1.4	Reference Goals.	13

1.2 Introduction

The purpose of this document is to detail the hardware and software required for a remote key entry (RKE) application. For several reasons this solution can fit well even for “wireless” applications where there is a need to send data from a remote sensor via radio frequency (RF) communication.

If you look at networking connections, some years ago the only way to implement these was with hundreds of meter of wires. Today we are faced with a new challenge. For clarification, some of the standards are highlighted below along with their main application.

1. **2.4-GHz Standards** — The main application for this standard is business wireless local area networks (LANs), where the speed of the communication is a key factor (up to 24-Mbps)
2. **Bluetooth** —Originally, Bluetooth was being studied to provide a solution to point-to-point short-range links for voice applications, such as cell phones to PDAs or a hands free automobile adapter kit that eliminates the need for cables and adapter sockets. Even in these cases, the purpose of the standard is to provide a high-speed solution for communicating huge quantities of data.

3. **Radio Frequency (RF)** — The number of RF applications are unlimited. An example of RF use would be with gate opening applications. RF can be considered as a cost-effective solution for small to medium volume data transmission (not at high speeds). For such applications, frequency bands most used are 200 MHz, 315 MHz, 433 MHz, 868 MHz, or 900 MHz. Some of the markets interested in this standard include appliances, smoke sensors, and home networking.

The RF standard is the subject of this document.

1.3 Reference Board

Over the past several years, Motorola has developed many RF semiconductor products including the MC68HC908RF2 microcontroller unit (MCU). This microcontroller includes the excellent performance of the HC08 Family along with an integrated RF transmitter.

Reference board features:

- Mother Board: Connectable to a PC through an RS232 connector for setting up the receiver. The setting can be implemented stand alone using the integrated encoder and a display unit on the board.
- Remote Transmitter: Implemented using an MC68HC908RF2 MCU with integrated Motorola technology for UHF transmitters. Eight buttons are implemented on the transmitter.
- Receiver Board: The MC3359x UHF is used to receive data from the transmitter. The received data can be viewed on the display unit on the mother board.
- 433-MHz and 868-MHz band frequency transmission implemented (the MC68HC908RF2 can transmit both of the frequencies by simply changing a pin status)
- Simple rolling code provided
- All source code and schematics provided for fast implementation of an RF application

It is important to highlight that the MC68HC908RF2 MCU can transmit at two frequency levels: 433-MHz or 868-MHz simply by changing the configuration of one pin (to be either a pullup or pulldown). For evaluation reasons, Motorola has provided two separate transmitters so that users can avoid buying or disconnecting a pullup on a case-by-case basis.

See [Figure 1-1](#).



Figure 1-1. Transmitter

1.4 Reference Goals

The main goal of this document is to demonstrate how to use the Motorola components listed in [1.3 Reference Board](#) to control transmitting and receiving a RF transmission. However, it is important to acknowledge the capabilities of the MC68HC908RF2's FLASH memory which will be demonstrated. A rolling code algorithm is provided with this material (the source code is in [Appendix A. Source Code](#)). This algorithm has been developed to use the embedded FLASH of the MC68HC908RF2 MCU instead of an EEPROM memory.

In the remote transmitter, eight buttons have been implemented. The pressure of one of these buttons is transformed in a RF transmitted frame and captured by the receiver. The display mounted on the mother board then shows an indication of the button pressed with the CRC code

transmitted (highlighting the fix code and variable code in case a rolling code transmission is chosen).

Due to the excellent current consumption features of the MC68HC908RF2 a “small” 3-volt Lithium battery (i.e., CR2032) is used in the remote transmitter. In this configuration the average transmission distance is around 10 meters. However, with a simple modification to the transmitter layout (including a small transistor for amplifying and a “bigger” battery) a range of 100 meters can be achieved.

The mother board is powered with a 9-volt DC power supply and has a encoder mounted to set up the configuration of the transmission (fixed or rolling code) and the three registers which reside inside the RF receiver. The value of these register can be viewed on the mother board’s display.

Another way to change the register’s value is through Windows® PC software which is provided on a CD ROM and downloadable from the Motorola web site at:

www.motorola.com/semiconductors

All embedded software has been written using Metrowerks® CodeWarrior® suite for Motorola HC08 devices. All the code is commented and provided in [Appendix A. Source Code](#).

The receiver board is shown in [Figure 1-2](#) and the mother board in [Figure 1-3](#).

® Windows is a registered trademark of Microsoft Corporation in the U.S. and/or other countries.

® Metrowerks and CodeWarrior are registered trademarks of Metrowerks, Inc., a wholly owned subsidiary of Motorola, Inc.



Figure 1-2. Receiver Board

General Description



Figure 1-3. Mother Board

Section 2. Motorola RF Components

2.1 Contents

2.2	Introduction	17
2.3	MC68HC908RF2	17
2.4	MC33592/3	19
2.5	MC33491	19

2.2 Introduction

This section provides a general overview of the Motorola RF components used in this reference. They are:

- MC68HC908RF2 — microcontroller unit (MCU)
- MC33592/3 — phase locked loop (PLL) tuned ultra-high frequency (UHF) receiver
- MC33491 — PLL tuned UHF transmitter

2.3 MC68HC908RF2

The MC68HC908RF2 MCU is a member of the low-cost, high-performance M68HC08 Family of 8-bit microcontroller units (MCUs). Optimized for low-power operation and available in a small 32-pin low-profile quad flat pack (LQFP), this MCU is well suited for remote keyless entry (RKE) transmitter designs.

This MCU can work with a Lithium battery because of the low level current consumption that can be reached. In stop mode, the MCU current consumption is almost 10 nA and it is possible to wakeup the MCU with an external interrupt generated by the keyboard interrupt port.

Features of the MC68HC908RF2 MCU include:

- High-performance M68HC08 architecture, fully upward-compatible object code with M6805, M146805, and M68HC05 Families.
- Maximum internal bus frequency of:
 - 4 MHz at 3.3 volts
 - 2 MHz at 1.8 volts
- Internal oscillator requiring no external components
- Software selectable bus frequencies
- ± 25 percent accuracy with trim capability to ± 2 percent
- Option allowing use of external clock source or external crystal/ceramic resonator
- 2 Kbytes of on-chip FLASH memory
- 128 bytes of on-chip RAM
- 16-bit, 2-channel timer interface module (TIM)
- 12 general-purpose input/output (I/O) ports:
 - Six shared with the keyboard wakeup function
 - Two shared with the timer module
- Port A pins have 3-mA sink capabilities
- Low-voltage inhibit (LVI) module:
 - 1.85-V detection forces MCU into reset
 - 2.0-V detection sets indicator flag
- 6-bit keyboard interrupt with wakeup feature
- External asynchronous interrupt pin with internal pullup (IRQ1)
- UHF MC33491 transmitter featuring:
 - Switchable frequency bands: 315, 434, and 868 MHz
 - On/off keying (OOK) and frequency shift keying (FSK) modulation
 - Adjustable output power range
 - Fully integrated voltage-controlled oscillator (VCO)
 - Data clock output for the MCU
 - Low external component count

2.4 MC33592/3

MC33592/3 is a PLL tuned UHF receiver, usable in the receiver side of a data transmitter application. It is a 24-lead quad flat package (LQFP) device, completely configurable by a serial peripheral interface (SPI) making it simple to interface with a MCU.

The MC33592/3 provides two band frequency ranges selectable by mask option:

- MC33592 supports 315/433 MHz band with an intermediate frequency (IF) of 500 kHz
- MC33593 supports 868 MHz band

The MC33592/3 supports either OOK or FSK modulation. With the data manager active, the MC33592/3 is able to receive Manchester codified data and communicate by SPI with an external MCU (a few components are externally requested). By using MC33592/3 in sleep mode with the strobe oscillator enabled, very low power consumption can be reached (115 mA). In this condition, MC33592/3 will be periodically woken up based on resistor-capacitor (RC) values outside and strobe ration programmed from users.

2.5 MC33491

MC33491 is a PLL tuned UHF transmitter in a 14-lead thin shrink small outline package (TSSOP). This device is characterized for low power consumption (0.5 nA stand-by current) with a supply voltage range between 1.9 volts and 3.7 volts (this feature is suitable for portable applications).

NOTE: *MC33491 can work at 315 MHz, 433 MHz, and 868 MHz without mask change as the frequency band is completely switchable. For the European market, 868 MHz will be a band extremely important in the future, since it has just recently become available in many countries.*

Refer to [Figure 2-1](#), [Figure 2-2](#), and [Table 2-1](#).

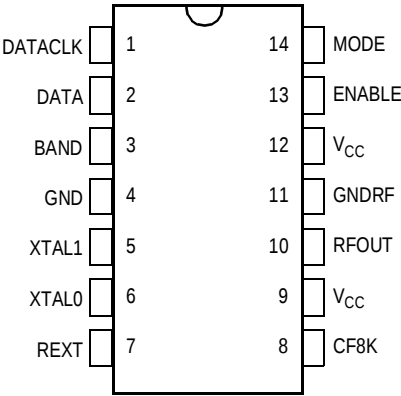


Figure 2-1. MC33492 Pin Assignments

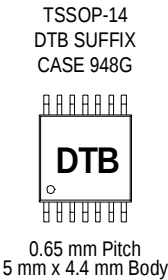


Figure 2-2. 14-Lead TSSOP Pictorial

Table 2-1. MC33492 Ordering Information

Device	Ambient Temperature Range	Package
MC33491 DTB	−40°C to 125°C	TSSOP14



Section 3. Reference Board Hardware

3.1 Contents

3.2 Introduction21
 3.3 Mother Board Description22
 3.4 Remote Description24
 3.5 Receiver Board Description24
 3.6 Transmitter28
 3.7 Mother Board Display Menu30

3.2 Introduction

This section describes the MC68HC908RF2 reference board hardware. The MC68HC908RF2 reference board (hereafter referred to as RF2 board) has these purposes:

- Demonstration of a remote controller board for the MC68HC908RF2 microcontroller (MCU)
- Demonstration of a receiver board for the MC3359X receiver chip

The RF2 board can also program the FLASH memory of a MC68HC908RF2 MCU from a personal computer (PC).

In the development of this reference:

- The software was developed using Metrowerks C compiler and assembler.
- Printed circuit board (PCB) and schematics were developed using ZUKEN-REDAC CR5000 software.
- Debugging was done using a MMEVS08 debugger with EML08GP32 daughter board for the MC68HC908GP32 and EML08RK2 daughter board for the MC68HC908RF2.

The RF2 board is composed of three parts:

1. Mother board — see [3.3 Mother Board Description](#)
2. Remote — see [3.4 Remote Description](#)
3. Receiver board — see [3.5 Receiver Board Description](#)

NOTE: *There is also a programming adapter board for the RF2 which is not covered by this document.*

3.3 Mother Board Description

The RF2 mother board is described here:

- The RF2 board is based on a MC68HC908GP32 FLASH MCU.
- The user interface has a rotary encoder with:
 - Push button
 - Select switch
 - 3-line by 12-character liquid crystal display (LCD)
- A RS232 interface is provided to connect the board to the PC.
- A programmable power supply generates all voltages required to program FLASH MCU.
- An input/output (I/O) connector is provided for the receiver board.

NOTE: *The receiver board has not been integrated with the mother board to allow for future upgrades as long as receiver chips are in continuous evolution.*

- Two option board connectors have been provided that can accept either a programming adapter or any other demonstration board.
- An optional external high density FLASH memory (128 Mbit) can be installed to use the RF2 board as a general-purpose FLASH programmer (firmware and adapters are not covered by this document).

Refer to **Figure 3-1** for a block diagram.

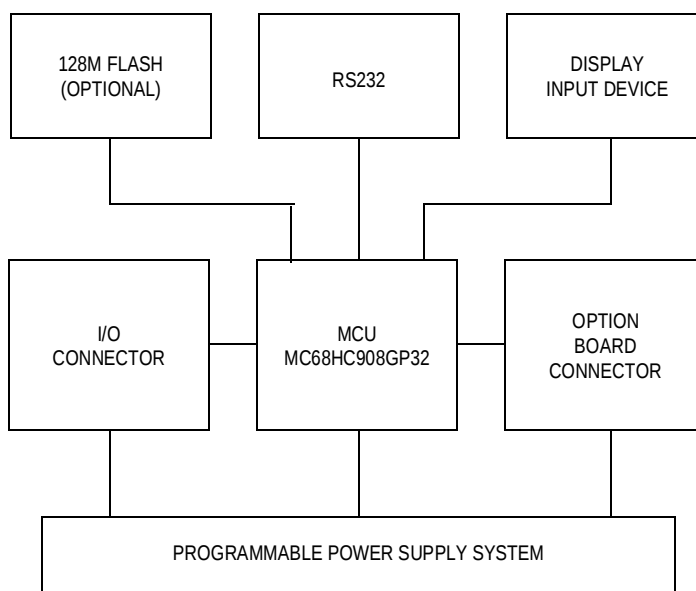


Figure 3-1. Mother Board Block Diagram

3.4 Remote Description

The remote is described here:

- The remote is based on a MC68HC908RF2 FLASH MCU with an integrated ultra-high frequency (UHF) module.
- The circuit has:
 - An 8-key silicon rubber keyboard to evaluate the wake-up feature of the MCU
 - A single cell CR2032 lithium battery
- The antenna is a small loop on the PCB generally used in keyless entry systems.

Refer to [Figure 3-2](#) for a block diagram.

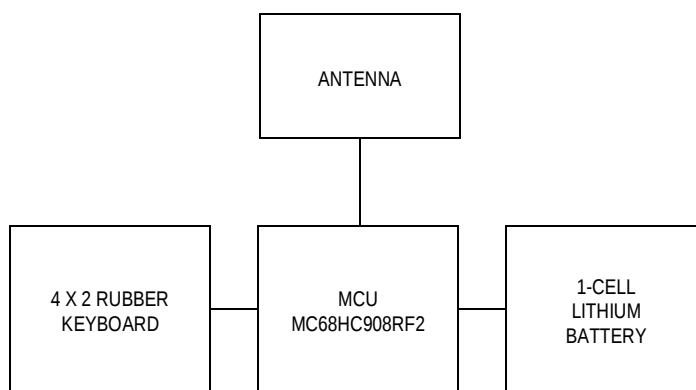


Figure 3-2. Remote Block Diagram

3.5 Receiver Board Description

The receiver board is based on a MC3359X UHF receiver (X = 1 and 2 for 433.92 MHz and X = 3 for 868 MHz). This circuit allows the demonstration of the MC3359X wake-up feature and data manager.

Two connectors are provided for expandability:

- One is a card edge connector so that the board can be used with the MC68HC908RF2 mother board
- The other (not assembled) is a standard 2-mm pitch PTH connector so that the board can be easily used with a user target.

Refer to [Figure 3-3](#) for a block diagram, [Figure 3-4](#) shows the receiver board, and [Figure 3-5](#) the receiver board schematic.

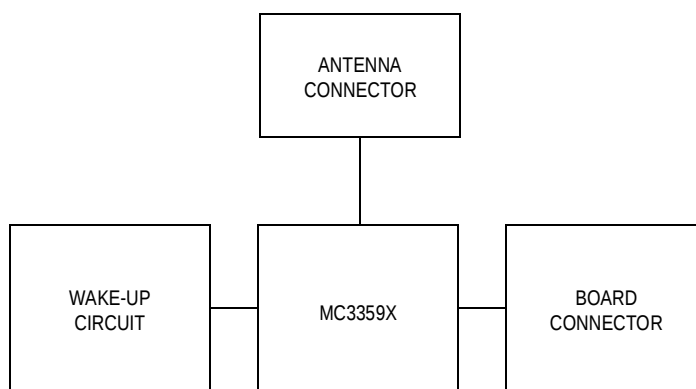


Figure 3-3. Receiver Board Block Diagram



Figure 3-4. Receiver Board

Reference Board Hardware

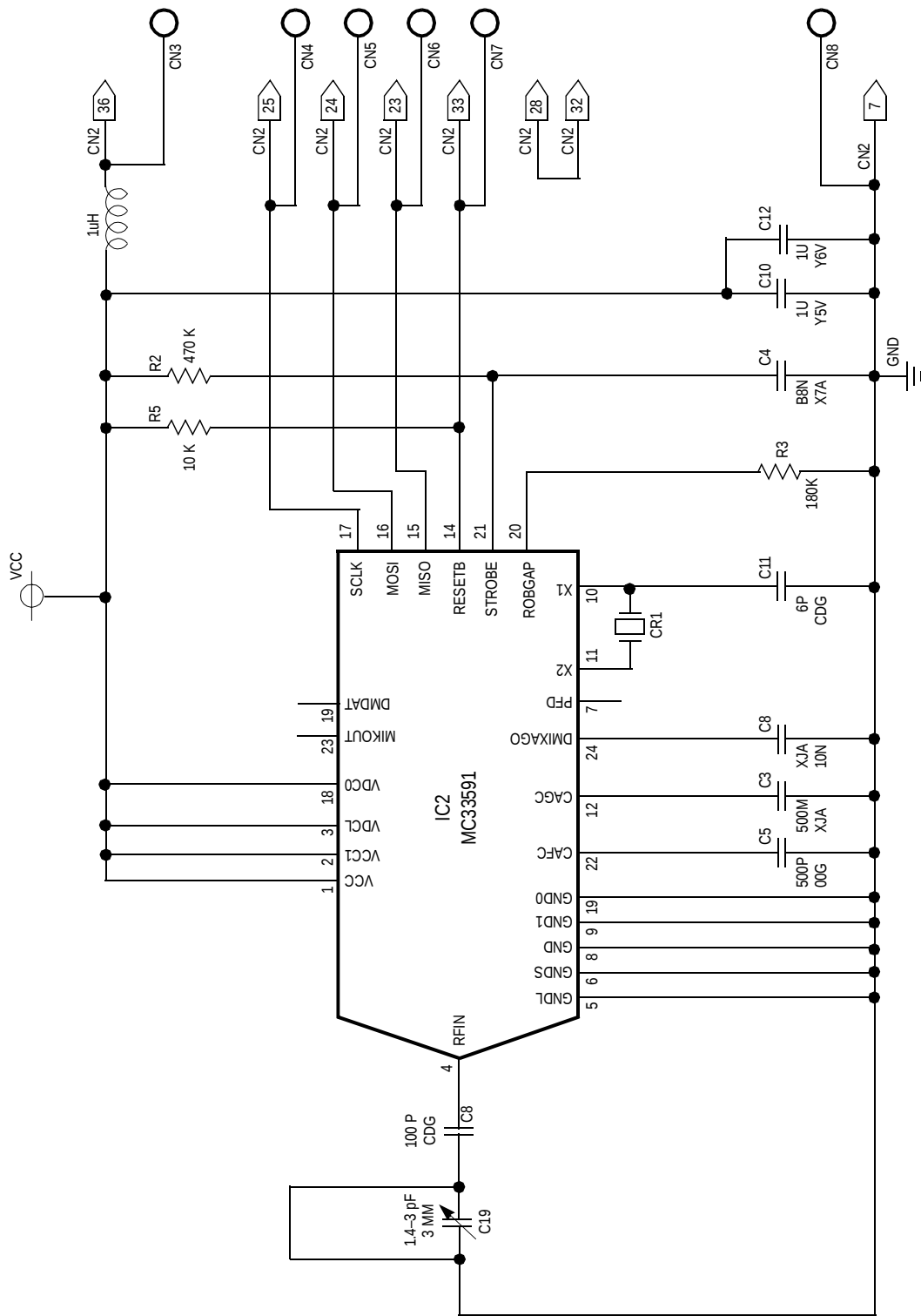


Figure 3-5. Receiver Board Schematic

The circuit is very simple and not all components are needed in most applications.

- The power supply is first filtered with L1, C10, and C12. Note that C10 must be placed as near as possible to IC2.
- R3 is the bandgap biasing resistor and should have an 1% tolerance.
- CR1 and C11 are the reference oscillators for the phase locked loop (PLL). Note that it is also possible to use an external generator connected to X1 through a DC blocking capacitor. However, the amplitude of the signal should not exceed 500 mV.

NOTE: When you calculate the reference oscillator frequency you must add to the carrier center frequency the IF center frequency (generally 660 kHz). So, for 433.92 Mhz carrier frequency:

$$f(\text{ref}) = (433.92 + 0.66) / 32 = 13.580625 \text{ MHz}$$

In this way, the transmitter and receiver have different reference XTAL.

WARNING: Reference oscillators values must be calculated accurately since a difference of 10 kHz = 0.01 MHz will force the receiver out of tune (for example, a XTAL of 13.57 MHz is not usable for a carrier of 433.92 MHz).

- C3 is the AGC capacitor in the case of OOK modulation. In case of FSK modulation, the voltage at this pin is used as reference.
- C4 and R2 are the strobe oscillator external components. If a strobe oscillator is not used R2 should be the jumper that connects the strobe pin to V_{CC}. Otherwise, MC33592/3 will be in sleep mode.

WARNING: If strobe = V_{CC}, bit 4 of CR1 (strobe oscillator enable) must be 0.

- The antenna is a very small loop antenna. While it is not the best solution in that it is not well matched with the input impedance of the LNA input amplifier, for most applications it is good enough and there are no additional costs. A better solution would require a matching network (with at least one inductor) and a input SAW filter which is very expensive.

3.6 Transmitter

A programming connector is provided for in-circuit programming of the MC68HC908RF2 and for loading custom user programs. The transmitter is shown in [Figure 3-6](#) and the schematic in [Figure 3-7](#)



Figure 3-6. Transmitter

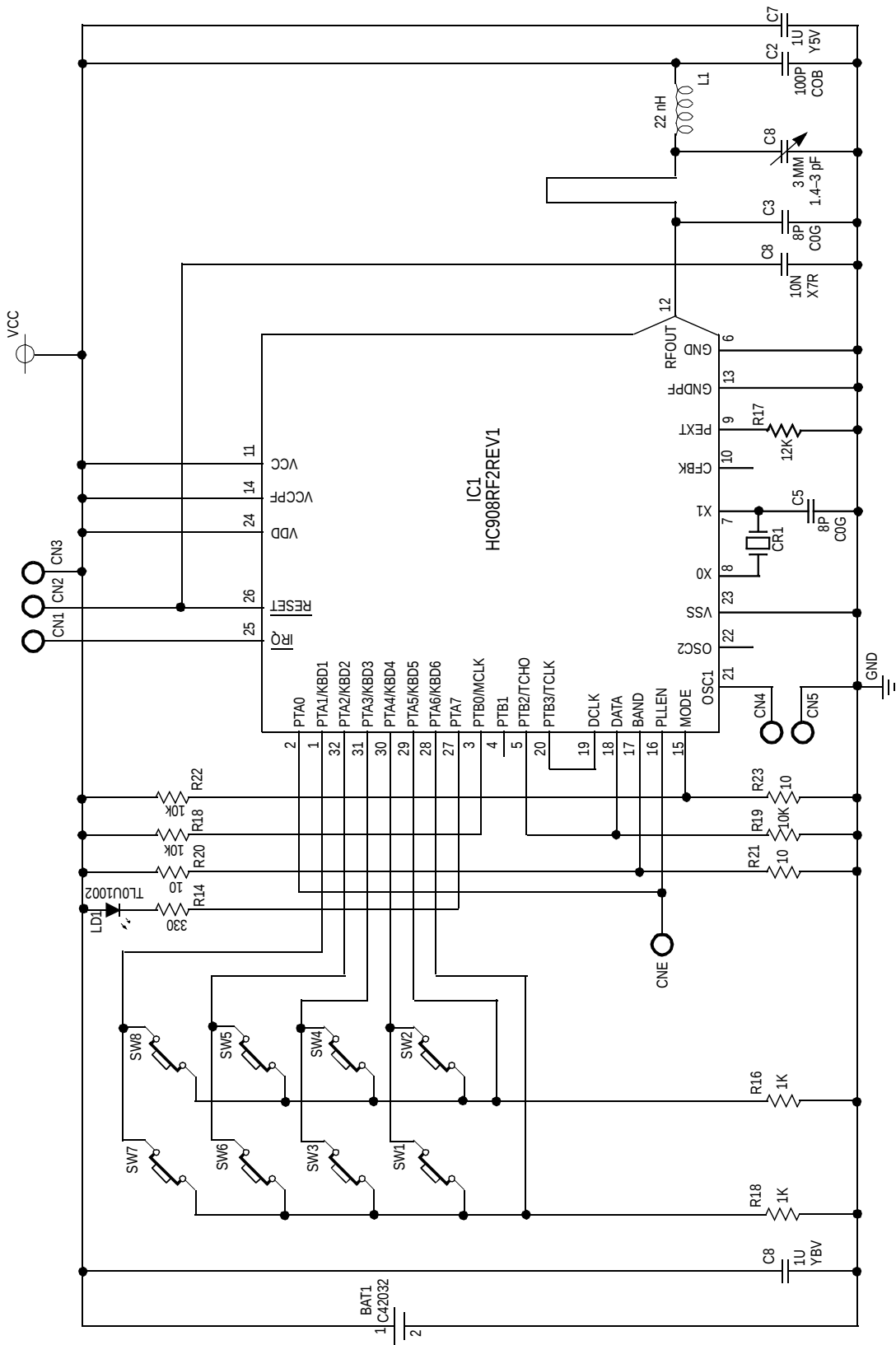


Figure 3-7. Transmitter Schematic

The circuit is very simple.

- The power supply is a CR2032 lithium battery connected directly to the MC68HC908RF2 (IC1).

NOTE: *When the RF PLL is disabled and the MCU is in sleep mode the power consumption is less than 100 nA.*

- The keyboard (a 4 x 2 crosspoint rubber keyboard) connected to the keyboard port wakes up the MCU.
- The MCU has an internal clock generator with no external components.
- CR1 and C6 are the reference oscillators for the RF PLL enabled by the PLEN pin. When PLEN = 1 the oscillator is ON and a reference signal DCLK = XTAL/64 is available. DCLK is used as a reference clock for the MCU timer to generate an accurate modulation signal. This signal is available only when PLL is ON.

WARNING: *The MC68HC908RF2 has two separate chips in the same case, one is the MCU and one is the RF transmitter. A buffer is needed if the MCU is to be used to drive other logic gates.*

- C2, C3, C8, C9, and L1 are the matching network for the small loop antenna.
- R20 and R21 are used to select the band and only one of them must be assembled.
- R12 is the output power setting resistor and sets the current of the RF power amplifier.

3.7 Mother Board Display Menu

Figure 3-8 shows the mother board display menu.

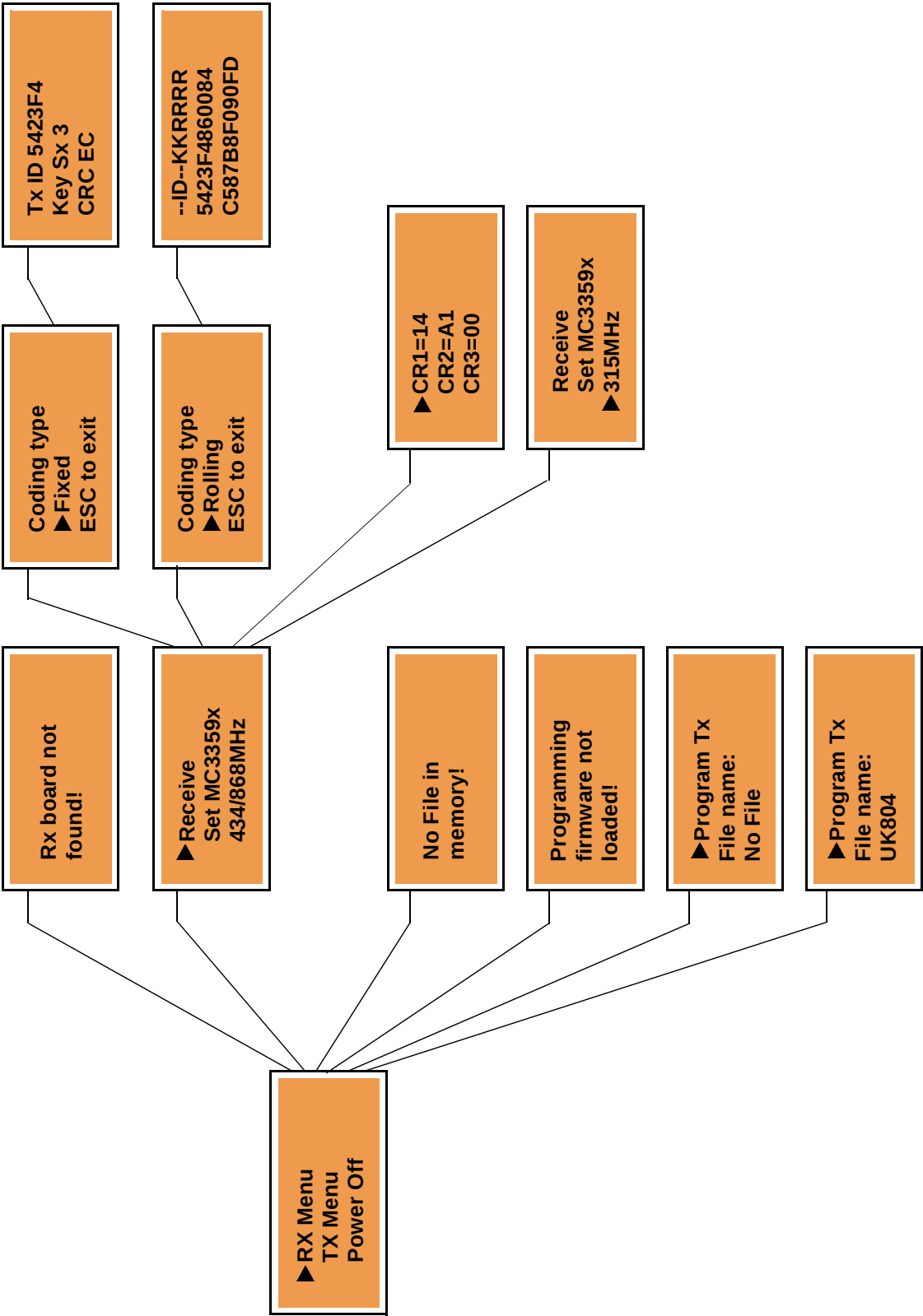


Figure 3-8. Mother Board Display Menu





Section 4. HC908RF2 for Windows Installation

4.1 Contents

4.2 Introduction33

4.3 Installation Screens33

4.2 Introduction

This section provides the installation screens for the MC68HC908RF2 reference board.

4.3 Installation Screens

See [Figure 4-1](#) through [Figure 4-8](#).

HC908RF2 for Windows Installation

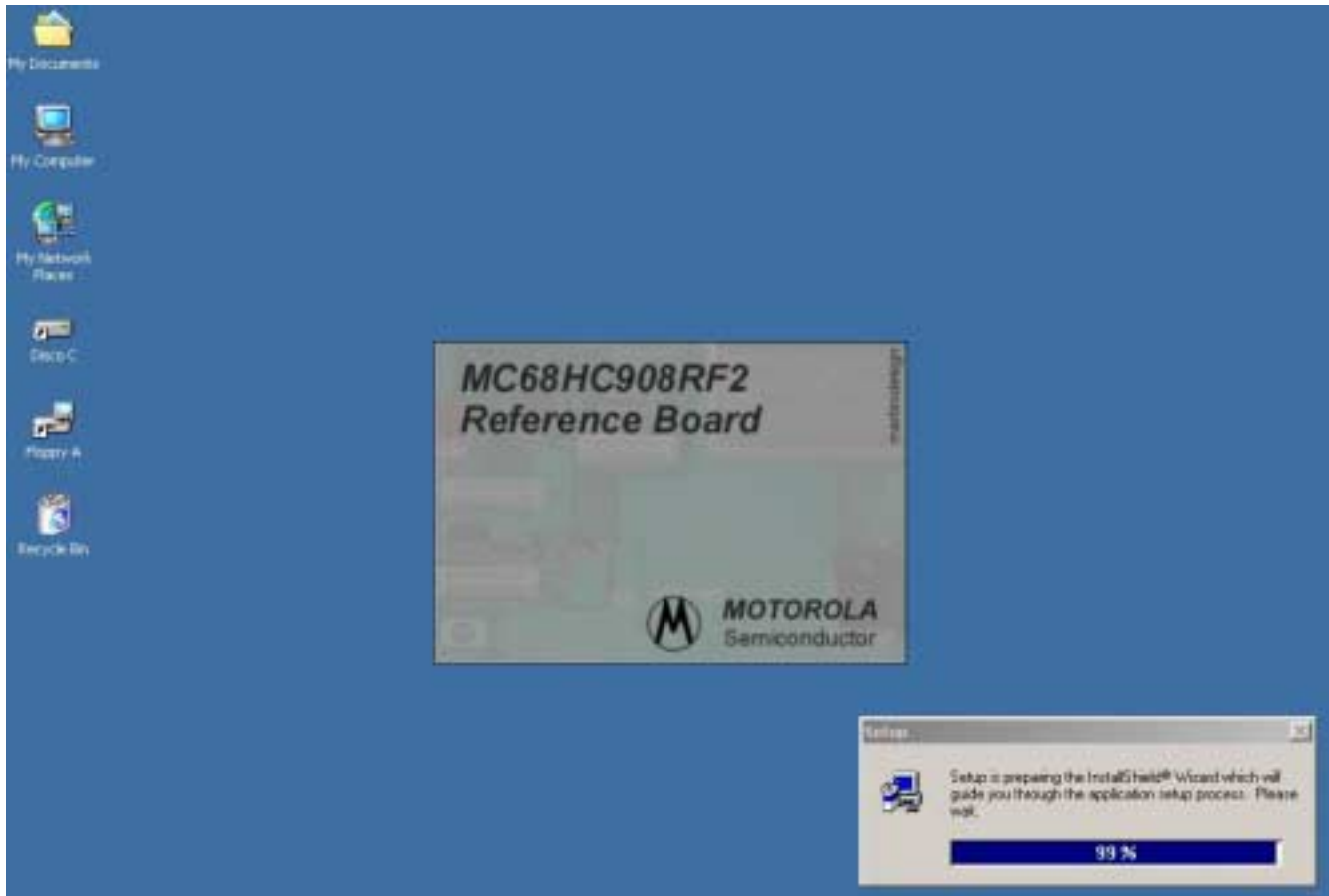


Figure 4-1. Startup Screen

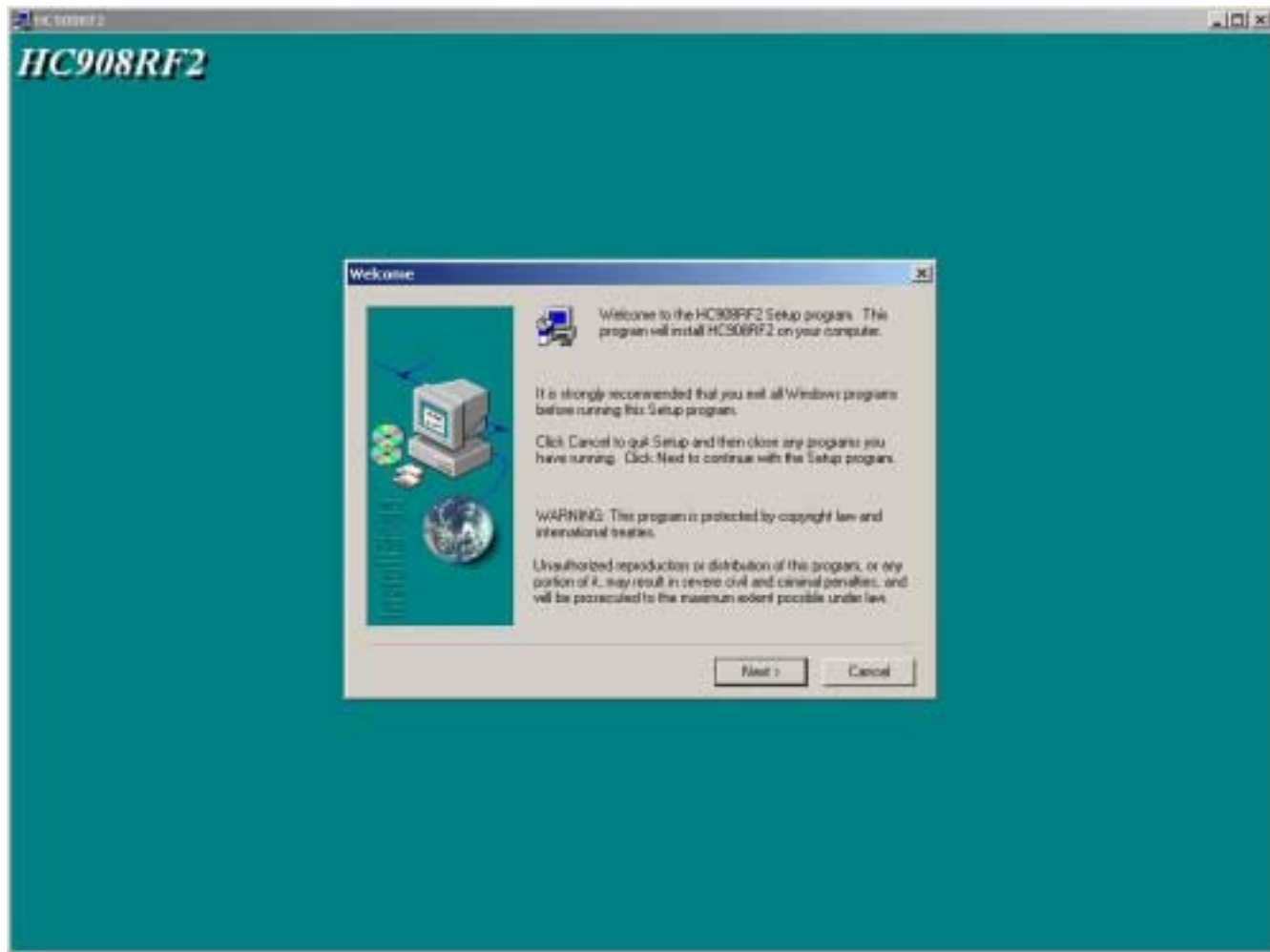


Figure 4-2. Welcome Screen

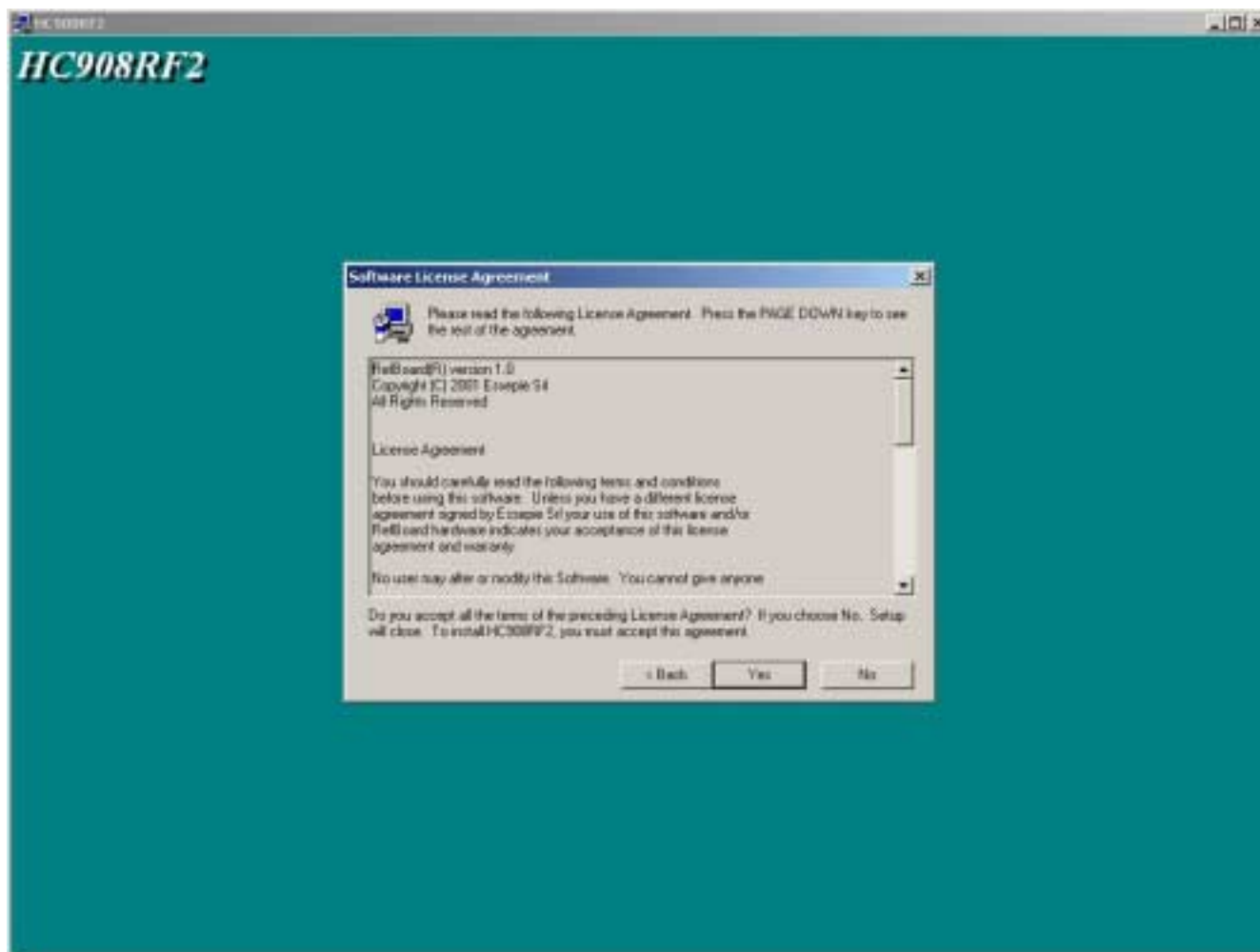


Figure 4-3. License Agreement

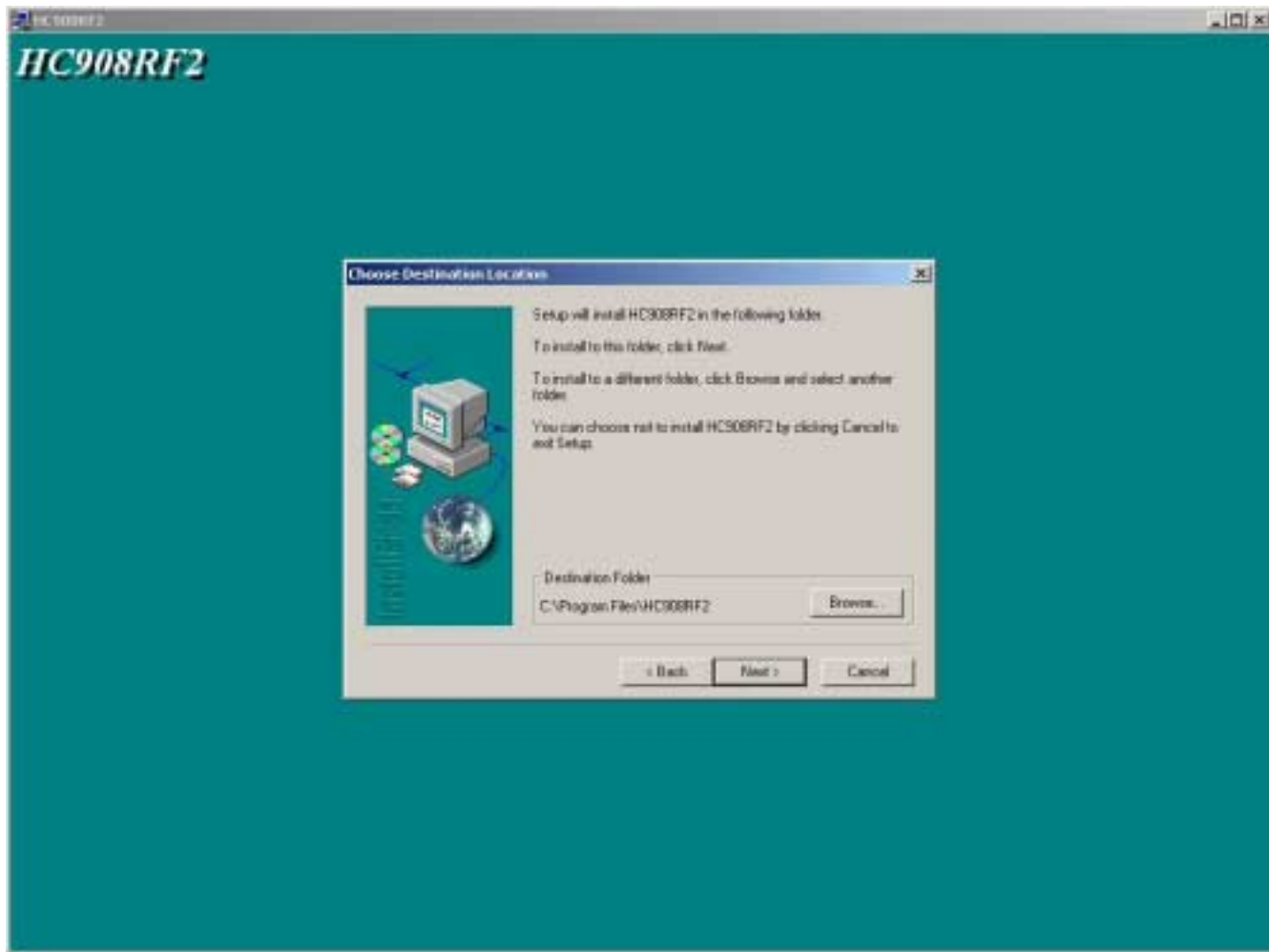


Figure 4-4. Choose Destination Location Screen

HC908RF2 for Windows Installation

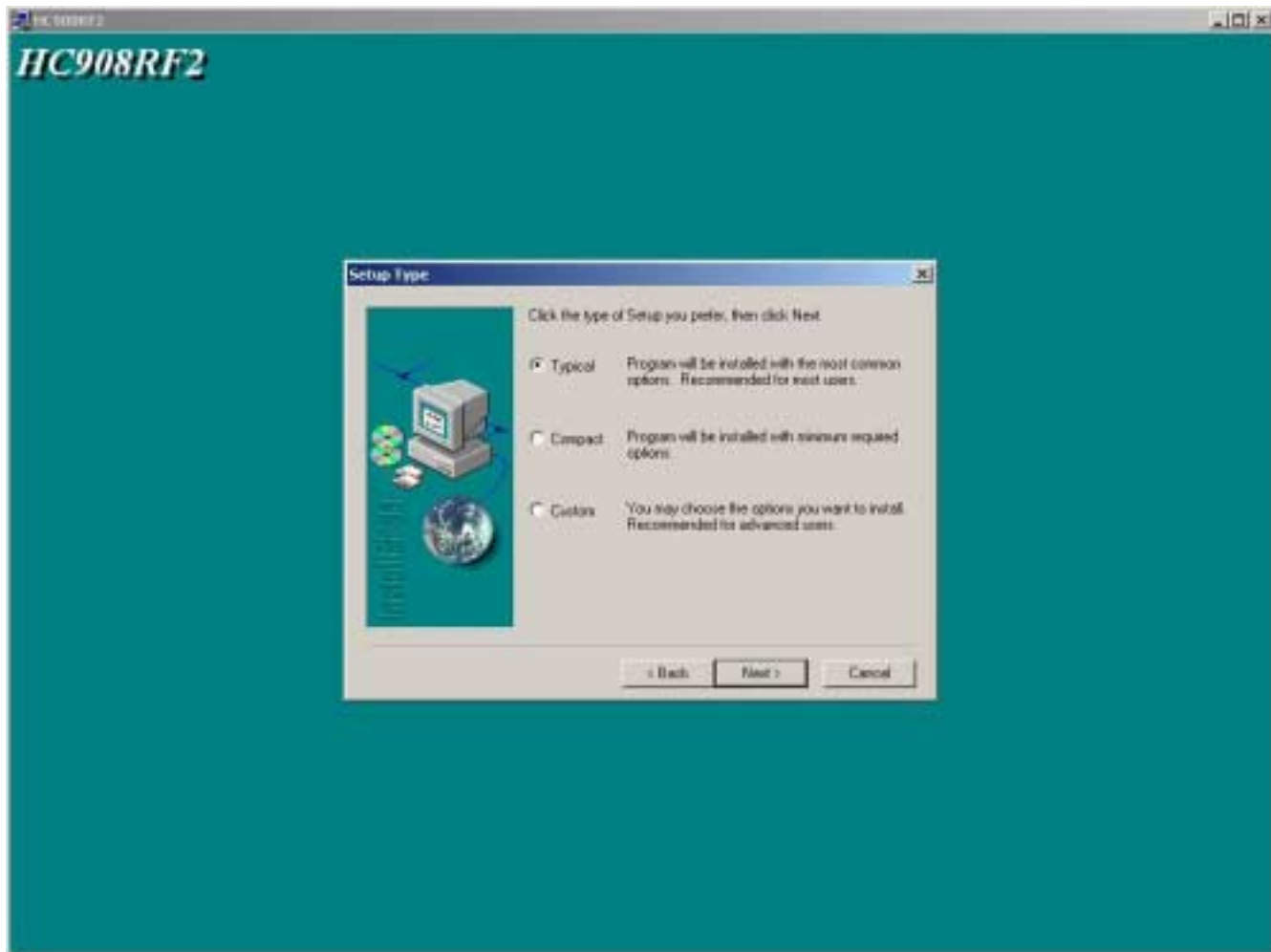


Figure 4-5. Setup Type

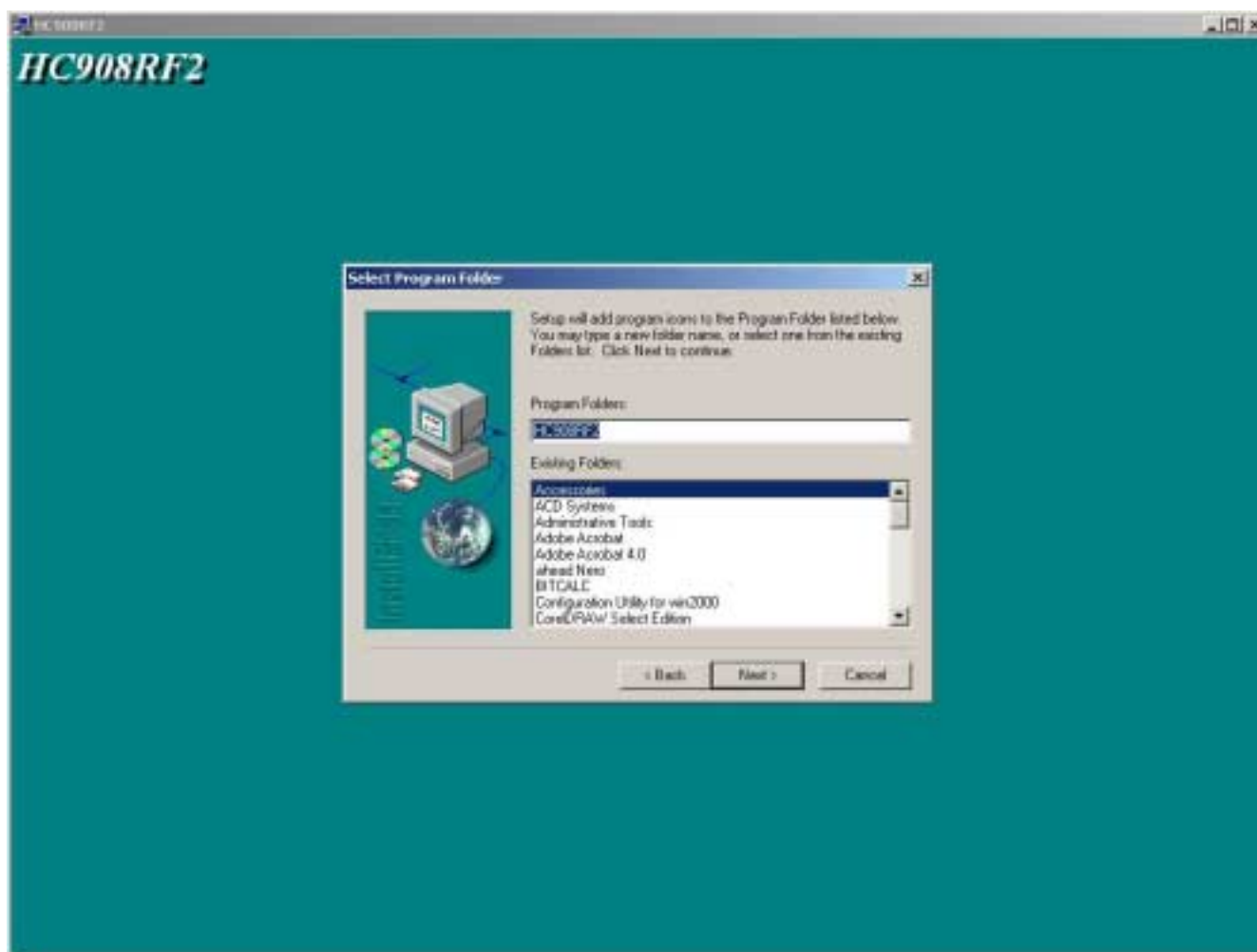


Figure 4-6. Select Program Folder

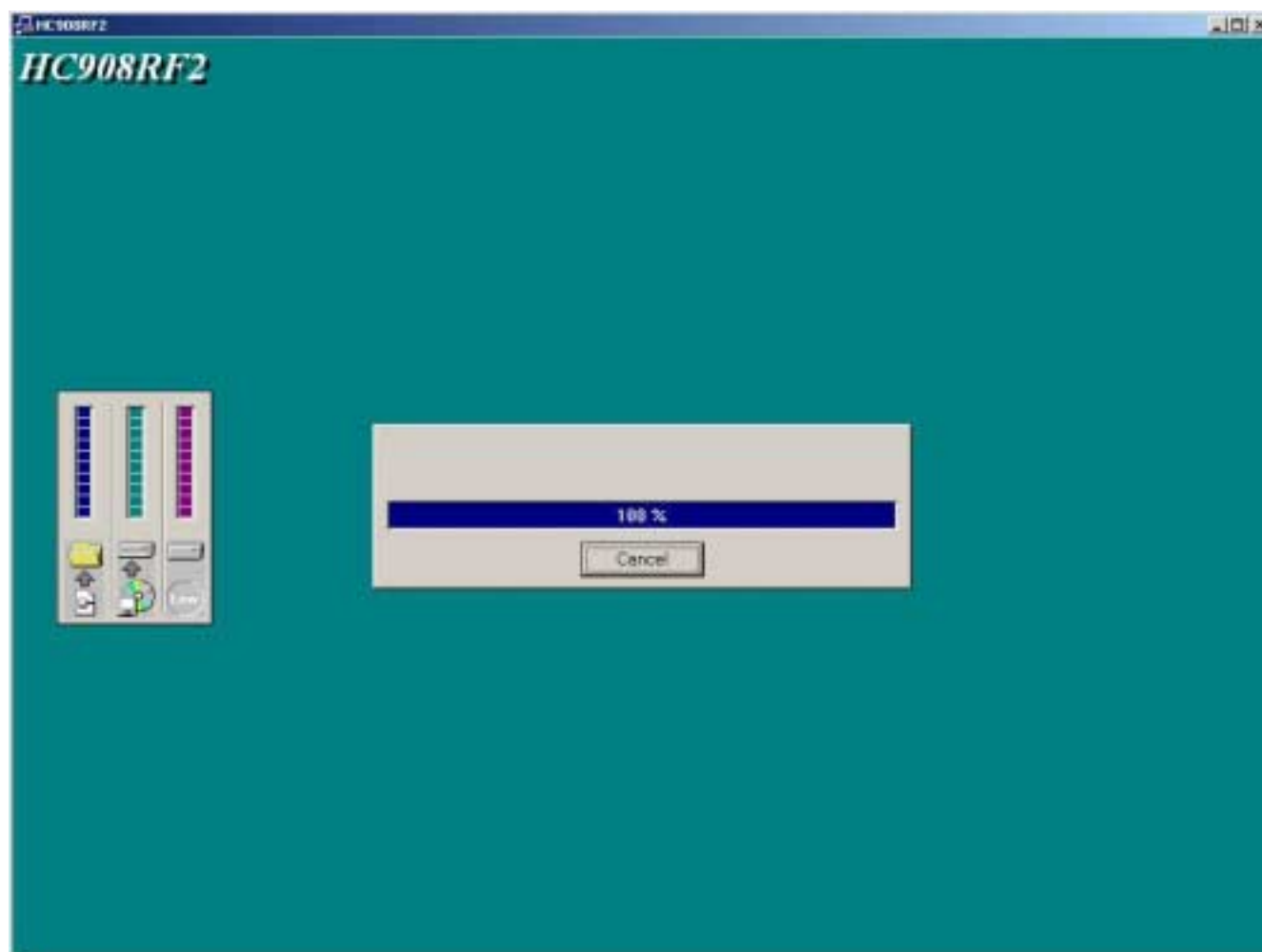


Figure 4-7. Install Files

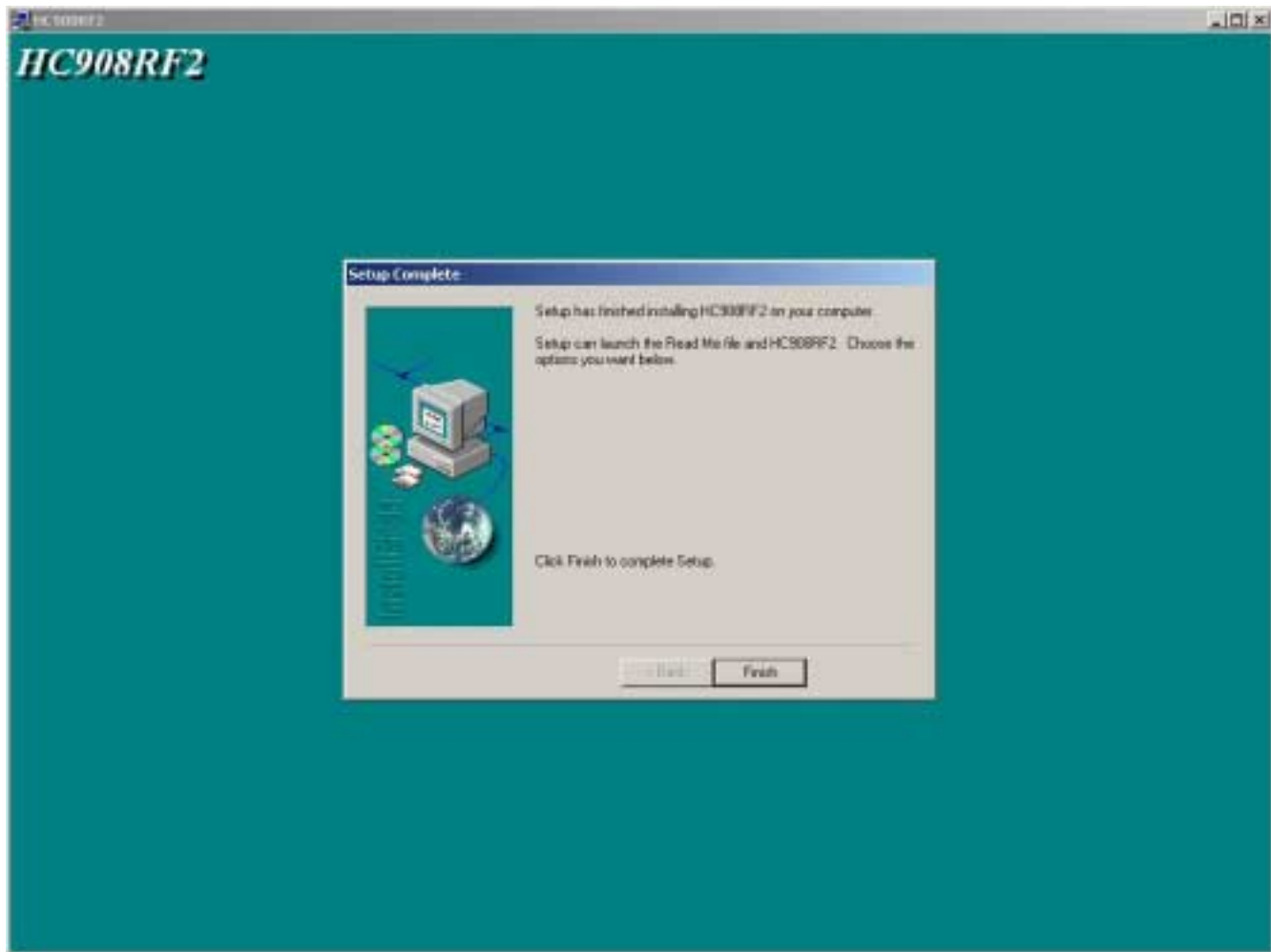


Figure 4-8. Finish



Section 5. Remote Programming Commands

5.1 Contents

5.2	Introduction	43
5.3	Communication Protocol	43
5.4	Command Set Parameters	44
5.4.1	Erase Programming Algorithm Memory (EPG)	45
5.4.2	Erase TX S19 Memory (ETX)	46
5.4.3	Erase User Code Memory (EUM)	47
5.4.4	Initialize MC33592/3 Registers (IRR)	48
5.4.5	Load Programming Algorithm (LPG)	49
5.4.6	Load TX S19 File (LTX)	51
5.4.7	Product ID (PID)	53
5.4.8	Read MC33592/3 Registers (RRR)	54
5.4.9	Update User Code (UUC)	55
5.4.10	MC33592/3 Registers (WRR)	57

5.2 Introduction

A set of commands is available to control the FlashProg from the RS232 serial interface. This section will explain the format and parameters of each command.

5.3 Communication Protocol

Commands are send to FlashProg using data packets. A data packet is composed of packet length, the command, parameters, data, and checksum. Parameters and data fields are optional.

Remote Programming Commands

The maximum length of a packet is 64 bytes. The packet length is 1 byte, the command is an ASCII mnemonic of three characters. Parameters and data are of variable size and the checksum length is 1 byte.

The command packet format is shown in **Figure 5-1**.

\$1B	PACKET LENGTH 1 BYTE	COMMAND 3 BYTES	PARAMETERS n BYTES	DATA m BYTES	CHECKSUM 1 BYTE
BYTE 0	BYTE 1				BYTE p

$$Checksum = \$FF - LSB \left(\sum_{k=p-1}^{k=1} Byte(k) \right)$$

Figure 5-1. Command Packet Format

Checksum is one's complement of the least significant bit (LSB) of the sum from packet length (included) to the checksum (excluded).

The answer from FlashProg can be one of four types:

- "Y" = Packet correct and command executed without error
- "N" = Packet error
- Exxx = Packet correct but command ended with error xxx
- A data packet

The answer data package format is shown in **Figure 5-2**.

\$1B	PACKET LENGTH 1 BYTE	DATA m BYTES	CHECKSUM 1 BYTE
BYTE 0	BYTE 1		BYTE p

$$Checksum = \$FF - LSB \left(\sum_{k=p-1}^{k=1} Byte(k) \right)$$

Figure 5-2. Command Packet Format

5.4 Command Set Parameters

This subsection provides the command set parameters available to control the FlashProg from the RS232 serial interface.

5.4.1 Erase Programming Algorithm Memory (EPG)

The EPG command will erase the memory area from \$D7FF to \$D6FF. The programming algorithm for the MC68HC908RF2 is stored in this area.

The EPG command packet has the following composition:

\$1B	\$04	EPG	\$1F
------	------	-----	------

Field	Value	Type	Length
Header	\$1B	Byte	1
Length	4	Byte	1
Mnemonic	EPG	Character	3
Checksum	Checksum	Byte	1

The answer to the EPG command has the following composition:

Answer	ASCII	Meaning
Y	\$59	Memory erased
N	\$4E	Packet not valid

Command examples:

To erase the programming algorithm area the following command packet must be sent.

\$1B,\$04,"EPG",\$1F

or in HEX

\$1B,\$04,\$45,\$50,\$47,\$1F

The answer will be:

Y

or in HEX

\$59

Remote Programming Commands

5.4.2 Erase TX S19 Memory (ETX)

The ETX command will erase the memory area from \$E700 to \$EFFF. The S19 file for the MC68HC908RF2 is stored in this area.

The ETX command packet has the following composition:

\$1B	\$04	ETX	\$0A
------	------	-----	------

Field	Value	Type	Length
Header	\$1B	Byte	1
Length	4	Byte	1
Mnemonic	ETX	Character	3
Checksum	Checksum	Byte	1

The answer to the ETX command has the following composition:

Answer	ASCII	Meaning
Y	\$59	Memory erased
N	\$4E	Packet not valid

Command examples:

To erase the programming algorithm area the following command packet must be sent.

\$1B,\$04,"ETX",\$0A

or in HEX

\$1B,\$04,\$45,\$54,\$58,\$0A

The answer will be:

Y

or in HEX

\$59

5.4.3 Erase User Code Memory (EUM)

The EUM command will erase the memory area from \$8000 to \$CFFF. The user code is stored in this area.

The EUM command packet has the following composition:

\$1B	\$04	EUM	\$14
------	------	-----	------

Field	Value	Type	Length
Header	\$1B	Byte	1
Length	4	Byte	1
Mnemonic	EUM	Character	3
Checksum	Checksum	Byte	1

The answer to the EUM command has the following composition:

Answer	ASCII	Meaning
Y	\$59	Memory erased
N	\$4E	Packet not valid

Command examples:

To erase the user code area the following command packet must be sent.

\$1B,\$04," EUM",\$14

or in HEX

\$1B,\$04,\$45,\$55,\$4D,\$14

The answer will be:

Y

or in HEX

\$59

Remote Programming Commands

5.4.4 Initialize MC33592/3 Registers (IRR)

The IRR command will load MC3359X registers CR1, CR2, and CR3 with the value stored in FLASH memory.

The IRR command packet has the following composition:

\$1B	\$04	IRR	\$0E
------	------	-----	------

Field	Value	Type	Length
Header	\$1B	Byte	1
Length	4	Byte	1
Mnemonic	IRR	Character	3
Checksum	Checksum	Byte	1

The answer to the EPG command has the following composition:

Answer	ASCII	Meaning
Y	\$59	Command executed successfully
N	\$4E	Packet not valid
E001	\$45, \$30, \$30, \$31	Receiver board not present

Command examples:

To INIT the MC3359X registers with the NV memory value the following command packet must be sent.

\$1B,\$04," IRR",\$0E

or in HEX

\$1B,\$04,\$49,\$52,\$52,\$0E

If the command has been executed successfully and is blank the answer will be:

Y

or in HEX

\$59

If the receiver board is not present the answer will be:

E001

or in HEX

\$45,\$30,\$30,\$31

5.4.5 Load Programming Algorithm (LPG)

The LPG command will load the programming algorithm for the MC68HC908GP32 and will wait for data.

The LPG command packet has the following composition:

\$1B	\$04	LPG	\$18
------	------	-----	------

Field	Value	Type	Length
Header	\$1B	Byte	1
Length	4	Byte	1
Mnemonic	LPG	Character	3
Checksum	Checksum	Byte	1

The answer to the LPG command has the following composition:

Answer	ASCII	Meaning
Y	\$59	Read to receive data
N	\$4E	Packet not valid

Command examples:

After the answer “Y”, the MC68HC908GP32 on the reference board is ready to receive data. The PC must send the character “S” and receive an “S” as echo.

The data is sent after the “S” and has the following composition:

Number of Packets (2 Bytes)	Packet 1		Packet N
-----------------------------	----------	-------	----------

Data format:

Number of Data Bytes (1 Byte)	Checksum (1 Byte)	Start Address	Data Bytes (Max 256)
-------------------------------	-------------------	---------------	----------------------

Remote Programming Commands

Packet format:

At the end of each packet the reference board will send the character "Y".

Command examples:

To start loading the programming algorithm area the following command packet must be sent:

\$1B,\$04,"LPG",\$18

or in HEX

\$1B,\$04,\$4C,\$50,\$47,\$18

The answer will be:

Y

or in HEX

\$59

5.4.6 Load TX S19 File (LTX)

The LTX command will load the programming algorithm for the MC68HC908GP32 and will wait for data. The data will be stored in the Tx S19 file area (0xE700 to 0xEFFF).

The LPG command packet has the following composition:

\$1B	\$04	LTX	\$03
------	------	-----	------

Field	Value	Type	Length
Header	\$1B	Byte	1
Length	4	Byte	1
Mnemonic	LTX	Character	3
Checksum	Checksum	Byte	1

The answer to the LTX command has the following composition:

Answer	ASCII	Meaning
Y	\$59	Ready to receive data
N	\$4E	Packet not valid

After the answer “Y”, the MC68HC908GP32 on the reference board is ready to receive data. The PC must send the character “S” and receive an “S” as echo.

The data is sent after the “S” and has the following composition:

Number of Packets (2 Bytes)	Packet 1		Packet N
-----------------------------	----------	-------	----------

Data format:

Number of Data Bytes (1 Byte)	Checksum (1 byte)	Start Address	Data Bytes (Max 256)
-------------------------------	-------------------	---------------	----------------------

Remote Programming Commands

Packet format:

At the end of each packet the reference board will send the character "Y".

Command examples:

To start loading the S19 file area the following command packet must be sent.

\$1B,\$04,"LTX",\$03

or in HEX

\$1B,\$04,\$4C,\$54,\$58,\$03

The answer will be:

Y

or in HEX

\$59

5.4.7 Product ID (PID)

The PID command will ask the MC68HCHC908RF2 for the firmware revision.

The PID command packet has the following composition:

\$1B	\$04	PID	\$1E
------	------	-----	------

Field	Value	Type	Length
Header	\$1B	Byte	1
Length	4	Byte	1
Mnemonic	PID	Character	3
Checksum	Checksum	Byte	1

The answer to the PID command has the following composition:

Answer	ASCII	Meaning
RFB XY.Z	\$52, \$46, \$42, \$20, \$XX, \$3Y, \$2E, \$3Z	
N	\$4E	Packet not valid

Command examples:

To get the firmware revision, the following command packet must be sent.

\$1B,\$04,"PID",\$1E

or in HEX

\$1B,\$04,\$50,\$49,\$44,\$1E

An example answer will be:

RFB A0.0

or in HEX

\$52,\$46,\$42,\$20,\$41,\$30,\$2E,\$30

Remote Programming Commands

5.4.8 Read MC33592/3 Registers (RRR)

The RRR command will read MC3359X registers CR1, CR2, and CR3.

The RRR command packet has the following composition:

\$1B	\$04	RRR	\$05
------	------	-----	------

Field	Value	Type	Length
Header	\$1B	Byte	1
Length	4	Byte	1
Mnemonic	RRR	Character	3
Checksum	Checksum	Byte	1

The answer to the RRR command has the following composition:

Answer	ASCII	Meaning
XYZ Y	\$XX, \$YY, \$ZZ, \$59	XX = CR1, YY = CR2, ZZ = CR3
N	\$4E	Packet not valid
E001	\$45, \$30, \$30, \$31	Receiver board not present

Command examples:

To read the MC3359X registers, the following command packet must be sent.

\$1B,\$04," RRR",\$05

or in HEX

\$1B,\$04,\$52,\$52,\$52,\$05

If the command has been executed successfully and CR1 = \$46, CR2 = \$04, CR3 = \$00 the answer will be:

\$46,\$04,\$00,Y

or in HEX

\$46,\$04,\$00,\$59

If the receiver board is not present the answer will be:

E001

or in HEX

\$45,\$30,\$30,\$31

5.4.9 Update User Code (UUC)

The UUC command will update the firmware stored in the area 0x8000 to 0xCFFF.

The UUC command packet has the following composition:

\$1B	\$04	UUC	\$0E
------	------	-----	------

Field	Value	Type	Length
Header	\$1B	Byte	1
Length	4	Byte	1
Mnemonic	UUC	Character	3
Checksum	Checksum	Byte	1

The answer to the UUC command has the following composition:

Answer	ASCII	Meaning
Y	\$59	Ready to receive data
N	\$4E	Packet not valid

After the answer “Y”, the MC68HC908GP32 on the reference board is ready to receive data. The PC must send the character “S” and receive an “S” as echo.

The data is sent after the “S” and has the following composition:

Number of Packets (2 Bytes)	Packet 1		Packet N
-----------------------------	----------	-------	----------

Data format:

Number of Data Bytes (1 Byte)	Checksum (1 Byte)	Start Address	Data Bytes (Max 256)
-------------------------------	-------------------	---------------	----------------------

Remote Programming Commands

Packet format:

At the end of each packet the reference board will send the character "Y".

Command examples:

To start updating the firmware the following command packet must be sent:

\$1B,\$04,"UUC",\$0E

or in HEX

\$1B,\$04,\$55,\$55,\$43,\$0E

The answer will be:

Y

or in HEX

\$59

5.4.10 MC33592/3 Registers (WRR)

The WRR command will write MC3359X registers CR1, CR2, and CR3 and update the INIT value stored in the FLASH memory.

The WRR command packet has the following composition:

\$1B	\$07	WRR	CR1, CR2, CR3 (3 Bytes)	Checksum
------	------	-----	----------------------------	----------

Field	Value	Type	Length
Header	\$1B	Byte	1
Length	\$07	Byte	1
Mnemonic	WRR	Character	3
Parameter	CR1, CR2, CR3	Byte	3
Checksum	Checksum	Byte	1

The answer to the WRR command has the following composition:

Answer	ASCII	Meaning
Y	\$59	Command executed successfully
N	\$4E	Packet not valid
E001	\$45, \$30, \$30, \$31	Receiver board not present

Remote Programming Commands

Command examples:

To set CR1 = \$54, CR2 = \$30, and CR3 = \$00 the following command packet must be sent:

\$1B,\$05,"WRR",\$54,\$30,\$00,\$79

or in HEX

\$1B,\$05,\$53,\$42,\$4D,\$03,\$79

The answer will be:

Y

or in HEX

\$59

If the receiver board is not present the answer will be

E001

or in HEX

\$45,\$30,\$30,\$31



Appendix A. Source Code

A.1 Contents

A.2 Mother Board Firmware60

A.2.1 UK509.ASM60

A.2.2 UK509 PRM124

A.3 Transmitter Firmware125

NOTE: The MC33592/3 is referred to as Romeo and the MC33491 is referred to as Tango in the source code.

Source Code

A.2 Mother Board Firmware

A.2.1 UK509.ASM

```

base $10

USERSUM      equ      $912D
USERSUM1     equ      $0569
USERSUM2     equ      $13EB

;Emulated    equ      0
;TestAgent   equ      0

;-----
;
; MC908RF2 reference board software
;
;-----
;
; SOURCE      : UK509.ASM
; CONTROLLER  : 68HC908GP32
; CLOCK       : 9.8304 Mhz
;
; VERSION     : B.00    04 Ago 2001
;
;-----
;
;          #$-VIEW:Schematics-$

Revision     equ      $0100
PlugVer      equ      3

include      "C:\Metrowerks\lib\HC08c\INC\H908GP32.INC"

;          #$-VIEW:Register Declarations-$$

;-----
;
; Port A
;
; Port A is used as 8 bit Data bus
;
;
; Pull Down : A(7)
;
;
DDRAInit     equ      %00000000      ;
PortAInit    equ      %00000000      ;

;-----
;

```

```

; Port B
;
; Port B bits
;
; CDE          : HN29F12814A Command Data Enable
; SC           : HN29F12814A Serial Clock
; OE           : HN29F12814A Output Enable
; Bank0        : Bank0 0/1 selct
; Bank1        : Bank1 0/1 selct
; WE           : HN29F12814A Write Enable
; SCLP         : I2C Bus Pod SCL
; SDAP         : I2C Bus Pod SDA
;
; Hardware changes
;
; None
;

; Porta B

CDE          equ      0
SC           equ      1
OE           equ      2
Bank0        equ      3
Bank1        equ      4
WE           equ      5
SCLP         equ      6
SDAP         equ      7

DDRInit      equ      %11111111
PortBInit    equ      %11111101

;-----
;
; Port C
;
; Port C bits
;
;KEY          : Select Switch : 0=Pushed 1=Not pushed
;EVHH         : Enable Mon08 VHH
;RST          : Mon08 Reset
;HOLD         : Main Power Supply Hold (1=Hold,0=Power Off)
;EVCC         : Enable Mon08 VCC
;NDAC         : Max533 DAC Enable (active low)
;LE           : Switch Led (0=On,1=Off)
;
; Hardware changes
;
; None

```

Source Code

```

; Porta C

KEY          equ      0          ;
EVHH         equ      1          ;
RST          equ      2          ;
HOLD         equ      3          ;
EVCC         equ      4          ;
NDAC         equ      5          ;
LED          equ      6          ;

DDRCInit     equ      %01111110  ;
PortCInit    equ      %01101000  ;

;-----
;
;
; Port D
;
; Port D bits
;
;Slave       : SPI Slave pin
;PodIRQ      : MON08 IRQ
;MOSI        : SPI MOSI
;PodData     : MON08 Data
;PUSHBUT     : Encoder Push Button (0=Pushed,1=Not pushed)
;ENCB        : Encoder Phase B
;SCL         : I2C Bus clock
;SDA         : I2C Bus data
;
; Hardware changes
;
; None
;

; Porta D

Slave        equ      0          ;
PodIRQ       equ      1          ;
MOSI         equ      2          ;
PodData      equ      3          ;
PUSHBUT      equ      4          ;
ENCB         equ      5          ;
SCL          equ      6          ;
SDA          equ      7          ;

DDRDInit     equ      %11000011  ;
PortDInit    equ      %11000101  ;

```

```
FlashRAM:      SECTION SHORT

LineLen        DS.B      1          ;
LineSum        DS.B      1          ;
StartAddr      DS.B      2          ;
EndAddr        DS.B      2          ;
ByteCount      DS.B      1          ;
NumLines       DS.B      2          ;
FlashTop       DS.B      2          ;

Dly30usPod     equ      $20          ;20us @ Fbus=2.4576MHz
```

```
BiosRAM:      SECTION SHORT
```

```
RFflags        DS.B      1
BufferRF       DS.B      8
BitCount       DS.B      1
RFCRC          DS.B      1
BufPtr         DS.B      1
LastTim        DS.W      1
Cell0          DS.B      1
DeltaT         DS.W      1
EncCnt         DS.B      1
EncStep        DS.B      1
CurX          DS.B      1
CurY          DS.B      1
Max            DS.B      1
Min            DS.B      1
CurItem       DS.B      1
CurMenuPtr    DS.B      2
TmpPtr         DS.B      2
RS232St        DS.B      1
CRC232         DS.B      1
SFlags         DS.B      1
BufferCnt      DS.B      1
SourcePtr      DS.B      2
DestPtr        DS.B      2
VHH            DS.B      1
RS232Tmo       DS.B      2
DispDump1      DS.B      14
EchoFlags      DS.B      1
LastV          DS.B      1
UpdateTmo      DS.B      1
MFlags         DS.B      1
CR1            DS.B      1
CR2            DS.B      1
CR3            DS.B      1
```

Source Code

```

BUFFERS:      SECTION

BufferSize    equ      $50                ;Size of RS232 buffer

RxBuffer      DS.B      50
StringBuffer  DS.B      18
StringBuffer1 DS.B      18
StringBuffer2 DS.B      18
FlashBuffer   DS.B      50

EEBuf         equ      StringBuffer

; Bit of RS232Status

EscRec        equ      1                ;Esc received (Receiving Packet)

; Bit of SFlags

UserC         equ      0                ;User commands enabled
TXFile        equ      1                ;1=Tx code present
UserP         equ      2                ;Programming code enabled

; Bit of MFlags

Cod           equ      0                ;0=Fixed,1=Rolling

ModeMsk       equ      %000000001      ;

; Bit of RFFlags

CRCP          equ      0                ;Flag CRC Pending
EndFrame      equ      1                ;Flag Lern OK
EdgRF         equ      2                ;Flag RF Edge
BlankW        equ      3                ;Flag Sync Waiting
LoadRF        equ      4                ;Flag Received Status
SyncW         equ      5                ;Flag Waiting Sync
Lockd         equ      6                ;Flag RF locked

RFMask        equ      %000000100      ;

```

```

;Bit of CR1

HE          equ      0          ;0=Write CRx
DME         equ      1          ;
SR0         equ      2          ;
SR1         equ      3          ;
SOE         equ      4          ;
MOD         equ      5          ;
CF          equ      6          ;
RW          equ      7          ;

CFMsk       equ      %01000000  ;

;-----
;
; Time contants

Dly5us      equ      $04        ;5uS @ FBus=2.4576MHz
Dly10us     equ      $08        ;10uS @ FBus=2.4576MHz
Dly30us     equ      $28        ;30uS @ FBus=2.4576MHz
Dly100us    equ      $51        ;100uS @ FBus=2.4576MHz

MinSync     equ      75         ;
MaxCell     equ      60         ;

;-----
;
;-----
;
; Blink Frequency divider
;
; This value is used to set the blinking frequency of the cursor during input

BlinkDiv    equ      %00001000

;-----

UserCode:   SECTION

UserCodeSum

DC.W        USERSUM
DC.W        UserCodeEnd

```

Source Code

UserProg

IFNDEF TestAgent

```
        jsr      Clr          ;
        jmp      MainMenu    ;
```

Warning

```
        pshx          ;
        pshh          ;
        jsr      Clr          ;
        pulh          ;
        pulx          ;
        jsr      PrintStr1    ;
        jsr      WaitInput    ;
        rts           ;
```

;Hex to string convesion

```
;
;On entry
; X,A=Pointer to string
; SP-3=Data
;On exit
; X,A=Pointer to end of string
```

HexToStr

```
        pshx          ;
        pulh          ;
        tax           ;
        lda          3,SP ;
        nsa          ;
        and          #$0F ;
        bsr          SetNibble ;
        lda          3,SP ;
        and          #$0F ;
        bsr          SetNibble ;
        clr          ,X   ;Null terminate string
        pshh          ;
        txa           ;
        pulx          ;
        rts           ;
```

SetNibble

```
        add          #$30 ;
        cmp          #$39 ;Convert to ascii
        bls          NoAlfa1 ;
        add          #$07 ;
```

NoAlfa1

```
        sta          ,X   ;
        aix          #$0001 ;
        rts           ;
```

DRM005

Remote Keyless Entry (RKE) Reference Design Using the MC68HC908RF2

;Char to Hex nibble

;

;On entry

; A=char

;On exit

; A=Hex value

CharToHex

sub #\$30 ;

cmp #\$09 ;

bls NibbleOK ;

sub #\$07 ;

NibbleOK

rts ;

;*****

;

; RS232 User routines

;

;*****

;*****

; Init Romeo Register Command

;

; Buffer status;

;

; 00 PackLen (Packet Length 04)

; 01-03 "IRR"

; 04 Packet Checksum (\$0E)

;*****

CmdIRR

jsr ChkRomeo ;

beq CmdIRR1 ;

lda #\$01 ;

jmp ErrCode ;

CmdIRR1

jsr InitRomeo ;

jmp CmdOK ;

;*****

;

Source Code

```

; Read Romeo Register Command
;
; Buffer status;
;
; 00      PackLen  (Packet Length 04)
; 01-03   "RRR"
; 04      Packet Checksum ($05)
; *****

CmdRRR
        jsr      ChkRomeo          ;
        beq      CmdRRR1          ;
        lda      #$01             ;
        jmp      ErrCode          ;

CmdRRR1
        jsr      ReadRomeo         ;
        lda      CR1              ;
        jsr      PutRS232          ;
        lda      CR2              ;
        jsr      PutRS232          ;
        lda      CR3              ;
        jsr      PutRS232          ;
        jmp      CmdOK            ;

; *****
; Write Romeo Registers Command
;
; Buffer status;
;
; 00      PackLen  (Packet Length 07)
; 01-03   "WRR"
; 04      CR1
; 05      CR2
; 06      CR3
; 07      Packet Checksum ($XX)
; *****

CmdWRR
        jsr      ChkRomeo          ;
        beq      CmdWRR1          ;
        lda      #$01             ;
        jmp      ErrCode          ;

CmdWRR1
        lda      RxBuffer+4        ;
        and      #$7F              ;
        sta      CR1              ;
        lda      RxBuffer+5        ;
        sta      CR2              ;
        lda      RxBuffer+6        ;
        sta      CR3              ;
        jsr      WriteRomeo        ;
        jsr      SaveNVCR          ;

```

CmdOK

```

lda    #'Y'                ;
jsr    PutRS232             ;
clr    BufferCnt            ;
clr    RS232St             ;
rts                    ;

```

```

;*****
;
; Send Error Code
;
; 01=No Romeo Module
;
;*****

```

ErrCode

```

psha                    ;
lda    #'E'                ;
jsr    PutRS232             ;
pula                    ;
jsr    SendDec              ;
clr    BufferCnt            ;
clr    RS232St             ;
rts                    ;

```

```

;*****
;
;
; ROME02 driver
;
;*****

```

ReadRomeo

```

pshh                    ;
pshx                    ;
psha                    ;
mov     #%00101010,SPCR    ;Init SPI as master
bclr    Bank0,PORTB        ;CS3 must be low
bclr    Bank1,PORTB        ;
lda     #$80                ;
jsr     SendRomeo           ;
jsr     GetRomeo            ;
sta     CR1                 ;
lda     #$00                ;
bsr     SendRomeo           ;
bsr     GetRomeo            ;
sta     CR2                 ;
lda     #$00                ;
bsr     SendRomeo           ;
bsr     GetRomeo            ;
sta     CR3                 ;

```

Source Code

```

        bset      Bank0, PORTB      ;CS3 must be high
        bset      Bank1, PORTB      ;
        pula      ;
        pulx      ;
        pulh      ;
        rts      ;

ChkRomeo
        jsr      ReadRomeo          ;
        lda      CR2                ;
        psha      ;
        lda      #$AA              ;
        sta      CR2                ;
        bsr      WriteRomeo         ;
        jsr      ReadRomeo          ;
        lda      CR2                ;
        ldx      1, SP              ;
        sta      1, SP              ;
        stx      CR2                ;
        bsr      WriteRomeo         ;
        pula      ;
        cmp      #$AA              ;
        rts      ;

InitRomeo
        lda      NVRegs            ;
        sta      CR1                ;
        lda      NVRegs+1          ;
        sta      CR2                ;
        lda      NVRegs+2          ;
        sta      CR3                ;
        bsr      WriteRomeo         ;
        jsr      ReadRomeo          ;
        mov      #%10001010, SPCR   ;Init SPI slave
        bset      Bank0, PORTB      ;CS2 must be low
        bclr      Bank1, PORTB      ;
        rts      ;

RomeoSetID
        jsr      ReadRomeo          ;
        sta      CR2                ;

WriteRomeo
        pshh      ;
        pshx      ;
        psha      ;
        mov      #%00101010, SPCR   ;Init SPI as master
        bclr      Bank0, PORTB      ;CS3 must be low
        bclr      Bank1, PORTB      ;
        lda      CR1                ;

```

```

        and        #$7F                ;WE bit low
        bsr        SendRomeo           ;
        bsr        GetRomeo            ;
        lda        CR2                  ;
        bsr        SendRomeo           ;
        bsr        GetRomeo            ;
        lda        CR3                  ;
        bsr        SendRomeo           ;
        bsr        GetRomeo            ;
        bset       Bank0,PORTB          ;CS3 must be high
        bset       Bank1,PORTB          ;
        pula       ;                    ;
        pulx       ;                    ;
        pulh       ;                    ;
        rts                          ;

SendRomeo
        brclr      SPTE,SPSCR,*         ;Wait transmitter empty
        sta        SPDR                 ;
        rts                          ;

GetRomeo
        brclr      SPRF,SPSCR,*         ;
        lda        SPDR                 ;
        rts                          ;

SPIProc
        tst        SPSCR                ;
        lda        SPDR                 ;
        rti                          ;

;*****
;
; Reception Routines without Data Manager Enabled
;
;*****

InitCode
        jsr        Home                 ;
        ldhx       #Frame1              ;
        brclr      Cod,MFlags,PrintFrame ;
        ldhx       #Frame2              ;

PrintFrame
        jsr        PrintStr1            ;

```

Source Code

InitRF

```
lda    #$08                ;
sta    BitCount            ;
lda    #$07                ;
sta    BufPtr              ;
bset   SyncW, RFflags      ;
bset   BlankW, RFflags     ;
bclr   Lockd, RFflags      ;
bclr   CRCP, RFflags       ;
clr    RFCRC               ;
rts
```

; Check of the RF signal edges

GetRF

```
brclr   EndFrame, RFflags, NoteEOF ;End Frame ?
jmp     EOF
```

NoteEOF

```
lda    PORTD                ;
eor     RFflags              ;Change in RF Input signal ?
and     #RFMask              ;
bne     NewEdge              ;
rts
```

; Calculation of DeltaT, Update of LastRTCC, Reset of Offset
; and change of edge

NewEdge

```
ldhx    T1CNTH              ;
;
; brclr   EdgRF, RFflags, NewEdge1 ;Rising edge ?
;
; lda     #$50              ;
; dbnza   $                 ;
; lda     PORTD             ;
; eor     RFflags           ;RF Input changed?
; and     #RFMask           ;
; bne     NewEdge1          ;
; rts
```

NewEdge1

```
txa
sub     LastTim+1            ;
sta     DeltaT+1            ;If Yes, determine DeltaT
pshh
pula
sbc     LastTim              ;
sta     DeltaT              ;
sthx    LastTim             ;
lda     #RFMask              ;Change of edge
eor     RFflags              ;
sta     RFflags              ;
brset   EdgRF, RFflags, RiseRF ;Rising edge ?
```

```
;Falling Edge: If don't wait for Sync determine the bit
```

```

ExitRF      bclr      LED,PORTC                ;
            bset      Lockd,RFflags            ;Synchronization Signal
            brclr     SyncW,RFflags,BitCalc    ;Sync waiting ?
            brset     BlankW,RFflags,ExitRF    ;Yes,Sync waiting ?
            ldhx      DeltaT                  ;Yes,check if Sync is correct
            cphx      #$0060                  ;
            bhi       InitRF                  ;
            cphx      #$0020                  ;
            blo       InitRF                  ;
            bclr      SyncW,RFflags            ;Sync flag reset

            rts                                ;Go out

```

```
;Rising Edge : Cell0=DeltaT
```

```

RiseRF      bset      LED,PORTC                ;
            brclr     Lockd,RFflags,ExitRF    ;Locked ?
            brclr     BlankW,RFflags,RiseRF1  ;Yes,Sync waiting ?
            ldhx      DeltaT                  ;Yes,check if Sync is correct
            cphx      #$0060                  ;
            bhi       InitRF                  ;
            cphx      #$0020                  ;
            blo       InitRF                  ;
            bclr      BlankW,RFflags          ;Sync Flag Reset
            rts                                ;

RiseRF1     lda       DeltaT                  ;
            bne       ExitRF                  ;
            lda       DeltaT+1                ;
            sta       Cell0                   ;
            rts                                ;

```

```
;Calcola bit
```

```

BitCalc     clrh                                ;
            lda       DeltaT                  ;Yes, check if the cell is correct
            bne       ExitRF                  ;
            lda       DeltaT+1                ;
            cmp       #MaxCell                ;
            bhi       FrameErr1              ;
            lda       Cell0                   ;Calculation of the Cell Rapport
            sub       DeltaT+1                ;
            ldx       BufPtr                  ;
            ror       BufferRF,X              ;bit Insertion
            brclr     CRCP,RFflags,NoCRC      ;CRC Pending ?
            lda       RFCRC                   ;

```

Source Code

```

        rora                ;
        ror      RFCRC      ;
        lda      RFCRC      ;
        eor      BufferRF,X  ;
        and      #%10000000 ;
        bclr     7,RFCRC    ;
        beq      NoCRC      ;
        bset     7,RFCRC    ;
        lda      #%00001100 ;
        eor      RFCRC      ;
        sta      RFCRC      ;

NoCRC

        dbnz     BitCount,ExitRF ;Next bit

        ;Next Byte

        lda      #$08       ;
        sta      BitCount   ;Yes, next byte
        dec      BufPtr     ;
        bset     CRCP,RFflags ;
        lda      BufPtr     ;
        bpl      ExitRF     ;
        bset     EndFrame,RFflags ;Yes,End Frame signal
        rts                ;No, go out

FrameErr1

        jmp      FrameErr   ;

        ; End Frame: Initialization and enable RF signal Research

EOF

        bclr     LED,PORTC  ;
        bclr     EndFrame,RFflags ;Yes,Reset End Frame
        lda      BufferRF+7 ;
        nsa      ;
        sta      BufferRF+7 ;
        lda      BufferRF+7 ;
        rora     ;
        rol      BufferRF+7 ;
        lda      RFCRC      ;Check CRC
        cmp      BufferRF+7 ;
        bne      FrameErr1  ;
        brclr    Cod,MFlags,NoRoll ;
        lda      #$00       ;
        ldx      #$02       ;
        jsr      GotoXY     ;
        lda      BufferRF+2  ;
        ldx      BufferRF+1  ;
        jsr      PrintShort ;
        lda      BufferRF+3  ;

```

```

        jsr    PrintHex          ;
        lda    BufferRF+4        ;
        jsr    PrintHex          ;
        lda    BufferRF+5        ;
        jsr    PrintHex          ;
        lda    BufferRF+7        ;
        jsr    PrintHex          ;
NoRoll
        ldhx   #BufferRF        ;
        lda    4,X               ;
        eor    5,X               ;
        sta    5,X               ;
        lda    3,X               ;
        eor    4,X               ;
        sta    4,X               ;
        lda    2,X               ;
        eor    3,X               ;
        sta    3,X               ;
        lda    1,X               ;
        eor    2,X               ;
        sta    2,X               ;
        lda    0,X               ;
        eor    1,X               ;
        sta    1,X               ;
        lda    BufferRF+6        ;
        and    #$0F              ;
        beq    NoRotate          ;
        tax                      ;
NextRotate
        lda    BufferRF          ;
        rola                   ;
        rol    BufferRF+5        ;
        rol    BufferRF+4        ;
        rol    BufferRF+3        ;
        rol    BufferRF+2        ;
        rol    BufferRF+1        ;
        rol    BufferRF          ;
        dbnzzx NextRotate        ;
NoRotate
        brclr  Cod,MFlags,ShowFix ;
ShowRoll
        lda    #$00              ;
        ldx    #$01              ;
        jsr    GotoXY            ;
        lda    BufferRF+2        ;
        ldx    BufferRF+1        ;
        jsr    PrintShort        ;
        lda    BufferRF+3        ;
        jsr    PrintHex          ;
        lda    BufferRF+4        ;
        jsr    PrintHex          ;

```

Source Code

```

        clra                ;
        jsr      PrintHex   ;
        lda      BufferRF+6  ;
        jsr      PrintHex   ;
        jmp      InitRF     ;

ShowFix
        lda      #$06       ;
        ldx      #$00       ;
        jsr      GotoXY     ;
        lda      BufferRF+2  ;
        ldx      BufferRF+1  ;
        jsr      PrintShort ;
        lda      BufferRF+3  ;
        jsr      PrintHex   ;
        lda      #$06       ;
        ldx      #$01       ;
        jsr      GotoXY     ;
        lda      BufferRF+4  ;
        bpl      KeyDx      ;
        lda      #'D'       ;
        jsr      PrintChar1 ;
        lda      #'x'       ;
        jsr      PrintChar1 ;
        lda      #' '       ;
        jsr      PrintChar1 ;
        bra      KeyNum     ;

KeyDx
        lda      #'S'       ;
        jsr      PrintChar1 ;
        lda      #'x'       ;
        jsr      PrintChar1 ;
        lda      #' '       ;
        jsr      PrintChar1 ;

KeyNum
        lda      BufferRF+4  ;
        and      #$0F       ;
        tax      ;
        lda      #$34       ;
        cbeqx    #$01, Key4  ;
        cbeqx    #$06, Key3  ;
        cbeqx    #$04, Key2  ;

Key1
        deca      ;

Key2
        deca      ;

Key3
        deca      ;

```

Key4

```

jsr    PrintChar1      ;
lda    #$06            ;
ldx    #$02            ;
jsr    GotoXY           ;
lda    BufferRF+5       ;
jsr    PrintHex         ;

```

nop

FrameErr

```

jmp    InitRF          ;

```

```

;-----
;
; Menu handling procedures
;
;-----

```

```

;-----
;
; Main menu routines
;
;-----

```

MainMenu

```

ldhx    #Menu1          ;
pshx    ;                ;
pshh    ;                ;
ldhx    #$0200          ;
clra    ;                ;
jsr     Menu             ;
ais     #$02             ;
bra     MainMenu         ;Endless loop on main menu

```

```

;-----
;
; Level 1 menu routines
;
;-----

```

RxMenu

```

jsr     ChkRomeo        ;
beq     RomeoFound      ;
jsr     Clr              ;
ldhx    #NoRomeoMsg      ;
jsr     PrintStr1        ;
jsr     WaitInput        ;
rts     ;                ;

```

Source Code

```

RomeoFound
    jsr    InitRomeo
    ldhx   #Menu11
    pshx
    pshh
    ldhx   #$0200
    clra
    jsr    Menu
    ais    #$02
    rts

TxMenu
    ldhx   #Menu12
    pshx
    pshh
    ldhx   #$0000
    clra
    jsr    Menu
    ais    #$02
    rts

;-----
;
; Level 1.1 menu routines
;
;-----

;Level 1.1 menu Do routines

DoReceive
    ldhx   #Menu11a
    pshx
    pshh
    ldhx   #$0101
    lda    #$01
    jsr    Menu
    ais    #$02
    jsr    WaitNotPush
    jsr    InitRomeo
    jsr    InitCode

RFLoop
    jsr    GetRF
    jsr    PushOK
    cbeqa  #$00, RFLoop
    bset   LED, PORTC
    rts

```

```

RomeoMenu
    ldhx    #Menu112        ;
    pshx                    ;
    pshh                    ;
    ldhx    #$0200          ;
    clra                    ;
    jsr     Menu            ;
    ais     #$02            ;
    rts                    ;

RxMode
    lda     #ModeMsk        ;
    eor     MFlags          ;
    sta     MFlags          ;
    rts                    ;

BandSel
    lda     #CFMsk          ;
    eor     CR1             ;
    sta     CR1             ;
    jmp     SaveNVCR        ;

Menu11aI2
    clra                    ;
    ldx     #$01            ;
    jsr     GotoXY          ;
    ldhx    #FixedMsg       ;
    brclr   Cod,MFlags,PrintMode ;
    ldhx    #RollMsg        ;

PrintMode
    jsr     PrintStr1       ;
    rts                    ;

Menu11I3
    clra                    ;
    ldx     #$02            ;
    jsr     GotoXY          ;
    ldhx    #Band315Msg     ;
    brclr   CF,CR1,PrintBand ;
    ldhx    #Band434Msg     ;

PrintBand
    jsr     PrintStr1       ;
    rts                    ;

```

Source Code

```

;-----
;
; Level 1.2 menu routines
;
;-----

;Level 1.1 menu Do routines

DoProgram
    brset    TXFile, SFlags, TX0K    ;
    jsr      Clr                     ;
    ldhx     #NoFileMsg1             ;
    jsr      PrintStr1               ;
    jsr      WaitInput               ;
    rts                                           ;

TX0K
    brset    UserP, SFlags, TX0K1    ;
    jsr      Clr                     ;
    ldhx     #BadProgMsg             ;
    jsr      PrintStr1               ;
    jsr      WaitInput               ;
    rts                                           ;

TX0K1
    jsr      ProgTx                  ;
    rts                                           ;

Menu12I3
    clra                                           ;
    ldx      #$02                     ;
    jsr      GotoXY                  ;
    bsr      ChkFileName             ;
    brset    TXFile, SFlags, PrintName ;

PrintNoName
    ldhx     #NoFileMsg              ;

PrintName
    jsr      PrintStr1               ;
    rts                                           ;

ChkFileName
    bclr     TXFile, SFlags           ;
    lda      FileName                ;
    cbeqa    #$FF, NoName            ;
    ldhx     #FileName               ;
    jsr      StrLen                   ;
    cmp      #$0C                     ;
    bhs      NoName                  ;
    bset     TXFile, SFlags           ;

NoName
    rts                                           ;

```

```

;-----
;
; Level 1.1.2 menu routines
;
;-----

EditCR1
    lda    CR1                ;
    clr    CurY               ;
    mov    #$05, CurX         ;
    jsr    InputByte          ;
    sta    CR1                ;
    bra    SaveNVCR           ;

EditCR2
    lda    CR2                ;
    mov    #$01, CurY         ;
    mov    #$05, CurX         ;
    jsr    InputByte          ;
    sta    CR2                ;
    bra    SaveNVCR           ;

EditCR3
    lda    CR3                ;
    mov    #$02, CurY         ;
    mov    #$05, CurX         ;
    jsr    InputByte          ;
    sta    CR3                ;

SaveNVCR
    lda    CR1                ;
    sta    FlashBuffer        ;
    lda    CR2                ;
    sta    FlashBuffer+1      ;
    lda    CR3                ;
    sta    FlashBuffer+2      ;
    ldhx   #NVRegs            ;
    sthx   SourcePtr          ;
    jsr    EraseFPAGE          ;Erase parameter flash page
    ldhx   #NVRegs            ;
    sthx   SourcePtr          ;
    aix    #$03                ;
    sthx   DestPtr            ;
    jsr    ProgFRange          ;Program parameter flash page
    jsr    InitRomeo           ;
    rts                        ;

;Level 1.2 draw routines

```

Source Code

```

Menu112I1
    jsr    ReadRomeo    ;
    lda    #$05         ;
    clrx                   ;
    jsr    GotoXY        ;
    lda    CR1           ;
    jsr    PrintHex      ;
    rts                   ;

Menu112I2
    jsr    ReadRomeo    ;
    lda    #$05         ;
    ldx    #$01         ;
    jsr    GotoXY        ;
    lda    CR2           ;
    jsr    PrintHex      ;
    rts                   ;

Menu112I3
    jsr    ReadRomeo    ;
    lda    #$05         ;
    ldx    #$02         ;
    jsr    GotoXY        ;
    lda    CR3           ;
    jsr    PrintHex      ;
    rts                   ;

NIL        equ    $0000

Menu1Msg    DC.B    " RX Menu    "
            DC.B    " TX Menu    "
            DC.B    " Power Off  ", $00
Menu11aMsg  DC.B    "Coding type "
            DC.B    " Fixed      "
            DC.B    "ESC to exit ", $00
Menu11Msg   DC.B    " Receive    "
            DC.B    " Set MC3359x ", $00
Menu12Msg   DC.B    " Program Tx  "
            DC.B    "File name:  ", $00
Menu112Msg  DC.B    " CR1=       "
            DC.B    " CR2=       "
            DC.B    " CR3=       ", $00
Band315Msg  DC.B    " 315MHz     ", $00
Band434Msg  DC.B    " 434/868MHz ", $00
FixedMsg    DC.B    " Fixed      ", $00
RollMsg     DC.B    " Rolling    ", $00
NoRomeoMsg  DC.B    "Rx board not "
            DC.B    "found !     ", $00
NoFileMsg   DC.B    "No File     ", $00
NoFileMsg1  DC.B    "No File in  "
            DC.B    "memory !    ", $00
BadProgMsg  DC.B    "Programming "
            DC.B    "firmware not "
            DC.B    "valid !     ", $00

```

```

;-----
;Menu 1 (main Menu)
;-----

Menu1          DC.W      Menu1Msg          ;Menu text
               DC.W      NIL                ;Draw Item 1
               DC.W      NIL                ;Draw Item 2
               DC.W      NIL                ;Draw Item 3
               DC.W      NIL                ;Draw Item 4
               DC.W      RxMenu             ;Do Item 1
               DC.W      TxMenu             ;Do Item 2
               DC.W      PowerOff           ;Do Item 3
               DC.W      NIL                ;Do Item 4

;-----
;Menu 1.1a (Coding Menu)
;-----

Menu11a        DC.W      Menu11aMsg        ;Menu text
               DC.W      NIL                ;Draw Item 1
               DC.W      Menu11aI2          ;Draw Item 2
               DC.W      NIL                ;Draw Item 3
               DC.W      NIL                ;Draw Item 4
               DC.W      NIL                ;Do Item 1
               DC.W      RxMode             ;Do Item 2
               DC.W      NIL                ;Do Item 3
               DC.W      NIL                ;Do Item 4

;-----
;Menu 1.1 (RX Menu)
;-----

Menu11         DC.W      Menu11Msg         ;Menu text
               DC.W      NIL                ;Draw Item 1
               DC.W      NIL                ;Draw Item 2
               DC.W      Menu11I3           ;Draw Item 3
               DC.W      NIL                ;Draw Item 4
               DC.W      DoReceive          ;Do Item 1
               DC.W      RomeoMenu          ;Do Item 2
               DC.W      BandSel            ;Do Item 3
               DC.W      NIL                ;Do Item 4

;-----
;Menu 1.1 (TX Menu)
;-----

Menu12         DC.W      Menu12Msg         ;Menu text
               DC.W      NIL                ;Draw Item 1
               DC.W      NIL                ;Draw Item 2
               DC.W      Menu12I3           ;Draw Item 3
               DC.W      NIL                ;Draw Item 4
               DC.W      DoProgram          ;Do Item 1
               DC.W      NIL                ;Do Item 2
               DC.W      NIL                ;Do Item 3
               DC.W      NIL                ;Do Item 4

```

Source Code

```

;-----
;Menu 1.1.2 (Romeo Menu)
;-----

Menu112      DC.W      Menu112Msg      ;Menu text
              DC.W      Menu112I1      ;Draw Item 1
              DC.W      Menu112I2      ;Draw Item 2
              DC.W      Menu112I3      ;Draw Item 3
              DC.W      NIL             ;Draw Item 4
              DC.W      EditCR1        ;Do Item 1
              DC.W      EditCR2        ;Do Item 2
              DC.W      EditCR3        ;Do Item 3
              DC.W      NIL             ;Do Item 4

Frame1      DC.B      "Tx ID      "
              DC.B      "Key      "
              DC.B      "CRC      ", $00
Frame2      DC.B      "--ID--KKRRRR"
              DC.B      "      "
              DC.B      "      ", $00

ENDIF

UserCodeEnd

;*****
;
; HC908RF2 Fast Programming Algorithm
;
;*****

ProgCode:      SECTION

ProgSum
              DC.W      USERSUM2
              DC.W      ProgTxEnd

ProgTx

IFDEF TestAgent

              jsr      Clr              ;
              ldhx     #NoProgMsg      ;
              jsr      PrintStr1        ;
              jsr      WaitInput        ;
              rts                    ;

NoProgMsg      DC.B      "Programming "
              DC.B      "firmware not "
              DC.B      "loaded !      ", $00

ENDIF

ProgTxEnd

```

```

;*****
;
; Remote user commands jump table
;
;*****

```

UserCmd: SECTION

```

UserCmdSum
    DC.W    USERSUM1
    DC.W    UserCmdTabEnd
UserCmdNum    DC.B    $03
UserCmdTab
    IFNDEF TestAgent
        DC.B    "IRR"
        DC.W    CmdIRR
        DC.B    "RRR"
        DC.W    CmdRRR
        DC.B    "WRR"
        DC.W    CmdWRR
    ENDIF
UserCmdTabEnd

```

TXCode: SECTION

```

FileName    DC.B    $FF

```

```

;-----
;
; Software del loader Agent rev A0.0
;
;-----

```

```

XDEF    LAInit
XDEF    LAIRQProc
XDEF    LAGetRS232

```

Loader: SECTION

```

LAInit
    mov    #%00000001, CONFIG1
    mov    #PortAInit, PORTA
    mov    #DDRAInit, DDRA
    mov    #PortBInit, PORTB
    mov    #DDRBBInit, DDRB

```

Source Code

```

mov      #PortCInit,PORTC      ;
mov      #DDRCInit,DDRC        ;
mov      #%00100001,PTCPUE     ;
mov      #DDRInit,DDR          ;
mov      #PortDInit,PORTD      ;
mov      #%00110000,PTDPUE     ;

;Clear RAM

ClrRAM   ldhx      #0040        ;

        clr       ,X           ;
        aix       #$01         ;
        cphx      #$0240       ;
        bne       ClrRAM       ;

        mov       #%01000000,SCC1 ;
        mov       #%00101100,SCC2 ;
        mov       #%00000010,SCBR ;
        mov       #%00000110,T1SCR ;
        mov       #%000000001,SPSCR ;Set SPI Baud rate
        ldhx      #$099A       ;Init Timer2 modulo register
        sthx      T2MODH       ;

LAWarmStart ldhx      #$0240        ;Init Stack pointer
        txs              ;

;init RS232 variables

        clr       BufferCnt      ;
        clr       RS232St       ;
        ldhx      #$0400       ;
        sthx      RS232Tmo      ;

;init Input device

        clr       EncCnt        ;
        mov       #$01,EncStep  ;
        clr       Min           ;
        clr       CurItem       ;

;init Output device

        jsr       LAMSetContrast ;
        jsr       LAInit7123    ;

;init VHH

        clr       VHH           ;
        jsr       LAMSetVHH     ;
        jsr       LAWait200ms   ;
        ldhx      #DevName      ;Startup screen

```

```

        jsr      LAPrintStr1          ;
        jsr      LAWait1s            ;
        jsr      LAClr               ;
        ldhx     #AgentMsg           ;BLIC Agent screen
        jsr      LAPrintStr1          ;
        jsr      LAWait1s            ;
        jsr      LAClr               ;
        ldhx     #SumMsg             ;BLIC Agent screen
        jsr      LAPrintStr1          ;

;Test User Command Table

        mov      #$02,CurY           ;
        lda      UserCmdSum+2        ;
        sta      TmpPtr              ;
        lda      UserCmdSum+3        ;
        sta      TmpPtr+1            ;
        ldhx     #UserCmdTab         ;
        jsr      LACalcUserSum        ;
        cmp      UserCmdSum+1        ;
        bne      LABadUserCmd        ;
        cpx      UserCmdSum          ;
        bne      LABadUserCmd        ;
        lda      #' '               ;
        bset     UserC,SFlags        ;
        bra      LATestPG            ;
LABadUserCmd
        bclr     UserC,SFlags        ;
        lda      #'x'               ;

;Test Programming algorithm code

LATestPG
        jsr      LAPrintChar1        ;
        mov      #$01,CurY           ;
        lda      ProgSum+2           ;
        sta      TmpPtr              ;
        lda      ProgSum+3           ;
        sta      TmpPtr+1            ;
        ldhx     #ProgTx             ;
        jsr      LACalcUserSum        ;
        cmp      ProgSum+1           ;
        bne      LABadProg           ;
        cpx      ProgSum             ;
        bne      LABadProg           ;
        lda      #' '               ;
        bset     UserP,SFlags        ;
        bra      LATestUC            ;

```

Source Code

```

LABadProg
    bclr    UserP,SFlags      ;
    lda     #'x'              ;

;Test User Code

LATestUC
    jsr     LAPrintChar1      ;
    clr     CurY              ;
    lda     UserCodeSum+2     ;
    sta     TmpPtr            ;
    lda     UserCodeSum+3     ;
    sta     TmpPtr+1          ;
    ldhx    #UserProg         ;
    jsr     LACalcUserSum     ;
    cmp     UserCodeSum+1     ;
    bne     LABadUserCode     ;
    cpx     UserCodeSum       ;
    bne     LABadUserCode     ;
    jsr     LAWait1s          ;
    jmp     UserProg          ;

LABadUserCode
    lda     #'x'              ;
    jsr     LAPrintChar1      ;
    jsr     LAWait1s          ;

LAUpdateUM1
    jsr     LAEraseUM         ;

LAUpdateUM2
    jsr     LAClr             ;
    ldhx    #FirmUpdMsg       ;
    jsr     LAPrintStr1       ;
    jsr     LAWait1s          ;
    cli     ;                  ;
    bra     $                  ;Wait RS232 commands

;Calculate user checksum and print it
;
;On entry
; H,X=Start address
; TempPtr=EndAddress
;On exit
; X,A=Checksum

LACalcUserSum
    clra     ;
    psha     ;
    psha     ;

```

```

LANextCodeSum
    lda    ,X                ;
    add    2, SP             ;
    sta    2, SP             ;
    clra                   ;
    adc    1, SP             ;
    sta    1, SP             ;
    aix    #$01              ;
    cphx    TmpPtr           ;
    bne    LANextCodeSum    ;
    lda    #$06              ;
    ldx    CurY              ;
    jsr    LAGotoXY          ;
    ldx    1, SP             ;
    lda    2, SP             ;
    jsr    LAPrintShort      ;
    lda    #' '              ;
    jsr    LAPrintChar1      ;
    pulx                   ;
    pula                   ;
    rts                      ;

;Software reset
;

LASoftReset
    lda    $FFFE            ;
    psha                   ;
    pulh                   ;
    ldx    $FFFF            ;
    jmp    ,X               ;

;Power Off
;

LAPowerOff
    brclr   KEY,PORTC,*      ;Wait until button released
    bclr    HOLD,PORTC       ;Power Off
    bra     $                ;

;Delay of 1ms
;This delay is independent from PPL because Timer2 modulo
;is recalculated each time PLL is changed
;

```

Source Code

```

LADly1ms
    mov    #%00110000,T2SCR;Stop and reset Timer2
    bclr   7,T2SCR                ;Clear T0F
    bclr   5,T2SCR                ;Start Timer2
    brclr  7,T2SCR,*              ;Wait overflow
    bset   5,T2SCR                ;Stop Timer2
    rts                                ;

;Delay of N ms
;HX=N
;
LADlyNms
    jsr    LADly1ms                ;Delay of 1ms
    aix    #$FF                    ;
    cphx   #$0000                  ;
    bne    LADlyNms                ;
    rts                                ;

;Delay of 20ms
;
LAWait20ms
    pshx                                ;
    pshh                                ;
    ldhx   #$14                    ;
    bsr    LADlyNms                ;
    pulh                                ;
    pulx                                ;
    rts                                ;

;Delay of 200ms
;
LAWait200ms
    pshx                                ;
    pshh                                ;
    ldhx   #$C8                    ;
    bsr    LADlyNms                ;
    pulh                                ;
    pulx                                ;
    rts                                ;

;Delay of 1s
;
LAWait1s
    pshx                                ;
    pshh                                ;
    ldhx   #$3E8                   ;
    bsr    LADlyNms                ;
    pulh                                ;
    pulx                                ;
    rts                                ;

;Delay of 5us
;

```

```

LAWait5uS                                ;15+4*Dly5us including JSR
    pshx                                ;[2] Save X
    ldx      #Dly5us                     ;[2]
    dbnzx    $                           ;[4*Dly5us]
    pulx                                ;[2] Restore X
    rts                                    ;[4]

;*****
;
; MAX533 DAC Device Driver
;
; MAX533 parameters
;
; X=DAC Control Byte 0000AACC
;           AA=DAC Number,CC=01=Load Inp.Register DAC AA
;           AA=DAC Number,CC=11=Load Inp.Register DAC AA and Load all DACs
;           AA=01,CC=00 Load all DACs
;
; A=DAC value
;
; Out0=Contrast control
; Out1=VMCU control voltage
; Out2=VHH control voltage
; Out3=Not used
;
;*****

; MAX533 Control Register Values

; MAX533

DACLCD      equ      %00000111
DACVHH      equ      %00000011

LASETContrast
    lda      Contrast                    ;
    ldx      #DACLCD                    ;Get Control Register Value
    bra      LAWriteDAC                 ;

LASETVHH
    lda      VHH                        ;
    ldx      #DACVHH                    ;

LAWriteDAC
    bclr     NDAC,PORTC                 ;CS Low for programming
    mov      #%00100000,SPCR            ;Init SPI as Master
    bset     SPE,SPCR                   ;Enable SPI
    stx      SPDR                       ;Send to SPI
    brclr    SPRF,SPSCR,*                ;Wait end of frame
    ldx      SPDR                       ;

```

Source Code

```

        brset    SPRF, SPSCR, *           ;Wait SPRF Low
        sta      SPDR                     ;Send second byte
        brclr    SPRF, SPSCR, *           ;Wait end of frame
        ldx      SPDR                     ;
        brset    SPRF, SPSCR, *           ;Wait SPRF Low
        bset     NDAC, PORTC              ;Rise CS to execute instruction
        bclr     SPE, SPCR                ;Disable SPI
        rts                                ;

```

```

;*****
;
; EA7123 LCD Device Driver
;
; Times to be respected:
; 1) t(BuF)>4.7us Bus Free Time (SDA=H SCL=H)
; 2) t(HdSta)> 4us Start Hold Time (SDA=L minimum time with SCL=H)
; 3) t(Low)>4.7us SCL Low Time
; 4) t(High)>4us SCL High Time
; 5) t(SuSto)> 4us Stop Setup Time (SCL=H minimum time before SDA rise)
;
;*****

```

```

EEAddr      equ      %10100000    ;24C00 I2C Bus Address

EA7123Addr   equ      %01110100    ;EA7123 I2C Bus Address
I2CRWmsk     equ      %00000001    ;I2C Bus R/W mask
EA7123WReg   equ      %00000000    ;Write EA7123 Redgister
EA7123RBusy  equ      %00100000    ;Read EA7123 Busy Flag
EA7123WData  equ      %01000000    ;Write EA7123 Data
EA7123RData  equ      %01100000    ;Read EA7123 Data

EA7123Clr    equ      %00000001    ;Read EA7123 Clear Display
EA7123Home   equ      %00000010    ;Read EA7123 Go Home
EA7123Mode   equ      %00000100    ;Read EA7123 Mode Set
EA7123Off    equ      %00001000    ;Read EA7123 Display Off
EA7123CursOff equ      %00001100    ;Read EA7123 Display On Cusror Off
EA7123CursOn  equ      %00001110    ;Read EA7123 Display On Cusror On Blink Off
EA7123BlinkOn equ      %00001111    ;Read EA7123 Display On Cusror On Blink On
EA7123CursMove equ      %00010100    ;Read EA7123 Cursor Move Right
EA7123Init    equ      %00101110    ;Read EA7123 Init Display Functions
                                12Charx4Row,Vlcd=V0-0.8

EA7123CAddr   equ      %01000000    ;Read EA7123 CGRAM Addr
EA7123DAddr   equ      %10000000    ;Read EA7123 DDARM Addr

LASendCtrlByte
        bclr     SDA, PORTD    ;SDA H->L

LASendByte
        ldx      #$08          ;

```

```

LANextBit
Time      jsr      LAWait5uS          ;Meet Start Hold Time and  SCL High
          bclr     SCL,PORTD          ;Change SDA with SCL Low
          bset     SDA,DDRD            ;SDA in Output
          lsla     ;Copy Bit 7 in Carry
          bcc      LASDALow           ;Copy Carry in SDA
          bset     SDA,PORTD          ;
          bra      LARiseSCL          ;
LASDALow
          bclr     SDA,PORTD          ;
LARiseSCL
          jsr      LAWait5uS          ;Meet SCL Low Time
          bset     SCL,PORTD          ;Rise SCL
          dbnzx    LANextBit          ;
          jsr      LAWait5uS          ;Meet SCL High Time
          bclr     SCL,PORTD          ;Controlla Ack
          bclr     SDA,DDRD            ;SDA in Input
          brn      $                  ;
          jsr      LAWait5uS          ;Meet SCL Low Time
          bset     SCL,PORTD          ;
          brclr    SDA,PORTD,LAExitByte ;Se Ack A=0
          lda      #$01               ;Se Nack A=1
LAExitByte
          rts                          ;
LARead7123
          lda      #(EA7123Addr+I2CRWmsk) ;Read EA7123
          jsr      LASendCtrlByte      ;
          cbeqa    #$01,LASStopBit     ;
LANextRead
          ldx      #$08                ;X as bit counter
LANextBitIn
          jsr      LAWait5uS          ;Meet SCL High Time
          bclr     SCL,PORTD          ;
          bclr     SDA,DDRD            ;SDA in Input
          jsr      LAWait5uS          ;Meet SCL Low Time
          bset     SCL,PORTD          ;
          clc                          ;
          brclr    SDA,PORTD,LASDALowIn ;
          sec                          ;
LASDALowIn
          rola     ;
          dbnzx    LANextBitIn         ;
          jsr      LAWait5uS          ;Meet SCL High Time
          bclr     SCL,PORTD          ;Ack out
          bset     SDA,DDRD            ;SDA in Output
          bclr     SDA,PORTD          ;
          jsr      LAWait5uS          ;Meet SCL Low Time
          bset     SCL,PORTD          ;
          rts                          ;

```

Source Code

```

LALastRead
LANextBitIn1    ldx        #$08                        ;X as bit counter
                jsr        LAWait5uS                    ;Meet SCL High Time
                bclr       SCL,PORTD                    ;
                bclr       SDA,DDRD                      ;SDA in Input
                jsr        LAWait5uS                    ;Meet SCL Low Time
                bset       SCL,PORTD                    ;
                clc                                           ;
                brclr      SDA,PORTD,LASDALowIn1          ;
                sec                                           ;
LASDALowIn1
                rola
                dbnzx    LANextBitIn1                    ;
                jsr        LAWait5uS                    ;Meet SCL High Time
                bclr       SCL,PORTD                    ;Nack out
                bset       SDA,DDRD                      ;SDA in Output
                bset       SDA,PORTD                    ;
                jsr        LAWait5uS                    ;Meet SCL Low Time
                bset       SCL,PORTD                    ;
                rts                                           ;

LAPutChar7123
                psha
                lda        #EA7123Addr                    ;Save Register value
                jsr        LASendCtrlByte                ;Send StartBit and Address
                cbeqa      #$01,LAError7123              ;
                lda        #EA7123WData                  ;If not Ack exit with error
                bra        LASendReg                      ;Send WriteData Control Register
                ;

LAWriteReg7123
                psha
                lda        #EA7123Addr                    ;Save Register value
                jsr        LASendCtrlByte                ;Send StartBit and Address
                cbeqa      #$01,LAError7123              ;
                lda        #EA7123WReg                   ;If not Ack exit with error
                ;Send WriteRegister Control Register

LASendReg
                jsr        LASendByte                    ;
                cbeqa      #$01,LAError7123              ;If not Ack exit with error
                pula
                jsr        LASendByte                    ;Recover Register value
                pshx
                ;Put it on the bus
                ;Dummy push

LAError7123
                pulx
                ;Pull out stacked value

LACStopBit
                bclr       SCL,PORTD                    ;Change SDA with SCL Low
                bclr       SDA,PORTD                    ;
                bset       SDA,DDRD                      ;SDA in Output
                bset       SCL,PORTD                    ;SDA in Output
                jsr        LAWait5uS                    ;Meet Stop Hold Time
                bset       SDA,PORTD                    ;
                jsr        LAWait5uS                    ;Meet Bus Free Time
                rts                                           ;

```

```

LAWaitBusy7123
    lda    #EA7123Addr          ;Send StartBit and Address
    jsr    LASendCtrlByte       ;
    cbeqa  #$01,LASStopBit      ;If not Ack exit with error
    lda    #EA7123RBusy         ;Send ReadBusyFlag Control Register
    jsr    LASendByte           ;
    cbeqa  #$01,LASStopBit      ;If not Ack exit with error
    bsr    LASStopBit           ;Send StopBit
    jsr    LARead7123           ;Read BusyFlag,first read is not

valid
LABusy7123
    jsr    LANextRead           ;Read BusyFlag
    bmi    LABusy7123           ;Busy Flag is in Bit7 (Bit7=1 LCD
busy)
    bsr    LALastRead           ;Last Read without aknowledge
    bra    LASStopBit           ;Send StopBit

LAHome
    clr    CurX                 ;
    clr    CurY                 ;
    lda    #EA7123Home          ;
    bra    LAWriteReg7123       ;

LAInit7123
    lda    #EA7123Init          ;
    bsr    LAWriteReg7123       ;
    cbeqa  #$01,LAEndInit       ;If not Ack exit with error
    lda    #EA7123CursMove      ;
    bsr    LAWriteReg7123       ;

LAClr
    clr    CurX                 ;
    clr    CurY                 ;
    lda    #EA7123Clr           ;
    bsr    LAWriteReg7123       ;
    cbeqa  #$01,LAEndInit       ;
    bsr    LAWaitBusy7123       ;
    lda    #EA7123CursOff       ;
    bra    LAWriteReg7123       ;

LAEndInit
    rts                        ;

LACursOn
    lda    #EA7123CursOn        ;
    jmp    LAWriteReg7123       ;

LACursOff
    lda    #EA7123CursOff       ;
    jmp    LAWriteReg7123       ;

```

Source Code

```

LABlinkOn
    lda    #EA7123BlinkOn    ;
    jmp    LAWriteReg7123    ;

LAGotoXY
    sta    CurX                ;Save X coordinate
    stx    CurY                ;
    psha                   ;
    txa                   ;
    and    #%00000011         ;Y coordinate modulo 4
    lsla                   ;
    nsa                   ;
    tax                   ;
    pula                   ;Restore X coordinate
    pshx                   ;Save Y coordinate
    ldhx    #$000E           ;
    div                   ;
    pshh                   ;H = X coordinate modulo 0x0E
    pula                   ;Move in A
    add    1,SP              ;Add Ycoordinate
    pulx                   ;Take it out of the stack
    add    #EA7123DAddr      ;Add register mask
    jmp    LAWriteReg7123    ;Set Cursor Address

LAPrintStr
    pshx                   ;Load pointer
    pulh                   ;
    tax                   ;

LAPrintStr1
    lda    ,X                ;
    beq    LAEndString       ;
    ora    #$80               ;
    aix    #1                 ;
    pshh                   ;Save pointer
    pshx                   ;
    bsr    LAPrintChar       ;
    pulx                   ;Restore pointer
    pulh                   ;
    bra    LAPrintStr1       ;

LAEndString
    rts                    ;

LAPrintChar1
    ora    #$80               ;

LAPrintChar
    jsr    LAPutChar7123     ;Print char
    inc    CurX               ;Increment cursor position
    lda    CurX               ;Check if new line
    clrh                   ;
    ldx    #$0C               ;
    div                   ;

```

```

    pshh                ;
    pulx                ;
    stx      CurX       ;
    cbeqa    #$00, LAEndPrintChar ;
    add      CurY       ;New line
    clrh                ;
    ldx      #$3        ;
    div                ;
    pshh                ;
    pulx                ;
    stx      CurY       ;
    lda      CurX       ;
    bsr      LAGotoXY   ;
LAEndPrintChar
    rts                ;

LAPrintDec
    ldx      #$0A       ;
    clrh                ;
    div                ;
    pshh                ;
    clrh                ;
    div                ;
    pshh                ;
    tsta                ;
    bne      LADigit3   ;
    lda      #' '       ;
    jsr      LAPrintChar1 ;
    pula                ;
    tsta                ;
    bne      LADigit2   ;
    lda      #' '       ;
    jsr      LAPrintChar1 ;
    bra      LADigit1   ;
LADigit3
    add      #$30       ;
    jsr      LAPrintChar1 ;
    pula                ;
LADigit2
    add      #$30       ;
    jsr      LAPrintChar1 ;
LADigit1
    pula                ;
    add      #$30       ;
    jsr      LAPrintChar1 ;
    rts                ;

LAPrintHex
    psha                ;
    pshx                ;
    bra      LAPrintLowByte ;

```

Source Code

```

LAPrintShort
    psha                ;Save Low byte
    pshx                ;Save High byte
    txa                 ;
    nsa                 ;
    and    #$0F         ;
    jsr    LAPrintNibble ;
    lda    1, SP        ;
    and    #$0F         ;
    jsr    LAPrintNibble ;

LAPrintLowByte
    lda    2, SP        ;
    nsa                 ;
    and    #$0F         ;
    jsr    LAPrintNibble ;
    lda    2, SP        ;
    and    #$0F         ;
    jsr    LAPrintNibble ;
    pulx                 ;
    pula                 ;
    rts                 ;

LAPrintNibble
    add    #$B0         ;
    cmp    #$B9         ;Convert to ascii
    bls    LNoAlfa     ;
    add    #$07         ;

LNoAlfa
    jsr    LAPrintChar  ;
    rts                 ;

LGetChar7123
    lda    #EA7123Addr   ;Send StartBit and Address
    jsr    LSendCtrlByte ;
    cbeqa  #$01, LStopBit1 ;If not Ack exit with error
    lda    #EA7123RData  ;Send ReadBusyFlag Control Register
    jsr    LSendByte     ;
    cbeqa  #$01, LStopBit1 ;If not Ack exit with error
    jsr    LStopBit      ;Send StopBit
    jsr    LRead7123     ;Read Data
    rts                 ;

LStopBit1
    jmp    LStopBit      ;

LInputHex
    psha                ;Space for local var
    psha                ;
    lda    Min           ;
    psha                ;Save Min
    lda    Max           ;

```

```

                psha                ;Save Max
                lda      EncCnt      ;
                psha                ;Save EncCnt
                lda      EncStep     ;
                psha                ;Save EncStep
                mov      #$30,Min     ;
                mov      #$46,Max     ;
                mov      #$FF,EncStep ;
                jsr      LAGetChar7123 ;
                sta      6,SP         ;Save Char
                sta      5,SP         ;Set Last Char
                sta      EncCnt       ;
                jsr      LASamePoint  ;
                jsr      LAWaitNotKeyOK ;
                jsr      LAWaitNotPush ;
                jsr      LASamePoint  ;
LAINputHexLoop
                bset     1,INTSCR     ;
                lda      EncCnt       ;
                cmp      #$3A         ;
                blo      LAInpHexOK   ;
                cmp      #$40         ;
                bhi      LAInpHexOK   ;
                lda      5,SP         ;
                cmp      EncCnt       ;
                bhi      LAChar39     ;
                lda      #$41         ;
                sta      EncCnt       ;
                bra      LAInpHexOK   ;
LACHar39
                lda      #$39         ;
                sta      EncCnt       ;
                ;
LAINpHexOK
                lda      5,SP         ;
                cbeq     EncCnt,LANoNewInp ;
                lda      EncCnt       ;
                bclr     1,INTSCR     ;
                sta      5,SP         ;
                jsr      LAPutChar7123 ;
                jsr      LASamePoint  ;
LANoNewInp
                bclr     1,INTSCR     ;
                jsr      LAPushOK      ;
                cbeqa    #$01,LAEndInpHex ;
                jsr      LAKeyOK      ;
                cbeqa    #$00,LAInputHexLoop ;
                jsr      LAGetChar7123 ;
                sta      6,SP         ;
                jsr      LASamePoint  ;

```

Source Code

```

LAEndInpHex
    pula
    sta    EncStep
    pula
    sta    EncCnt
    pula
    sta    Max
    pula
    sta    Min
    pula
    lda    1, SP
    jsr    LAPutChar7123
    jsr    LASamePoint
    pula
    sub    #$30
    cmp    #$0A
    blo    LAExitInpHex
    sub    #$07

LAExitInpHex
    rts

LASamePoint
    lda    CurX
    ldx    CurY
    jsr    LAGotoXY
    rts

;*****
;
; Menu manager
;
;*****

Arrow            equ        $2E                ;Menu selection char

;Redraw current menu
;
;On entry
; CurMenuPtr Ponter to menu template
; A=Current item
;On exit
; No parameter

LAMenuRedraw
    psha                                ;Save CurItem
    lda    #$04                        ;Init counter
    psha
    jsr    LAClr                        ;Clear screen
    ldhx   CurMenuPtr                  ;Get text pointer into HX
    lda    ,X

```

```

        psha                ;
        lda                1,X                ;
        psha                ;
        pulx                ;
        pulh                ;
        jsr                LAPrintStr1        ;Print menu mask
        ldhx               CurMenuPtr        ;
        aix                #$02                ;
LAMenuDrawLoop
        pshx                ;
        pshh                ;
        lda                ,X                ;Use DrawItem routine if pointer not
                                           null
        bne                LADrawItem         ;
        lda                1,X                ;
        beq                LANextItemDraw     ;
LADrawItem
        lda                ,X                ;
        psha                ;
        ldx                1,X                ;
        pulh                ;
        lda                4,SP               ;Get CurItem
        jsr                ,X                ;
LANextItemDraw
        pulh                ;
        pulx                ;
        aix                #$02                ;
        dbnz               1,SP,LAMenuDrawLoop ;
        clra                ;Print arrow at current item
        ldx                EncCnt            ;
        jsr                LAGotoXY          ;
        lda                #Arrow            ;
        jsr                LAPrintChar       ;
        pula                ;Free stack
        pula                ;
        rts                ;

;Menu Routine handles menu N
;
;On entry
; CurMenuPtr Ponter to menu template
; A=Current item
; H=Item Max
; X=Item min
;On exit
; No parameter

```

Source Code

```

LAMenu
    psha                ;Save CurItem
    pshx                ;Save Min
    pshh                ;Save Max
    sei                ;
    jsr    LAPushEncoder    ;Push Encoder status

LAInitMenu
    sei                ;
    lda    6, SP        ;
    sta    Min          ;
    lda    5, SP        ;
    sta    Max          ;
    mov    #$01, EncStep ;
    lda    $0A, SP      ;
    sta    CurMenuPtr   ;
    lda    $0B, SP      ;
    sta    CurMenuPtr+1 ;
    lda    7, SP        ;
    sta    EncCnt       ;
    bsr    LAMenuRedraw ;Draw menu (A=CurItem)
    cli    ;
    jsr    LAWaitNotKeyOK ;
    jsr    LAWaitNotPush ;

LASElectLoop
    jsr    LAKeyOK      ;
    cbeqa  #$01, LASelected ;If select do item action
    jsr    LAPushOK     ;
    cbeqa  #$01, LAEscaped ;Else exit menu
    jsr    LADly1ms     ;
    ldhx   RS232Tmo     ;
    cphx   #$0000       ;
    bne    LANoTmo      ;
    clr    RS232St      ;
    clr    BufferCnt     ;
    ldhx   #$0401       ;

LANoTmo
    aix    #$FF         ;
    sthx   RS232Tmo     ;
    lda    7, SP        ;
    cbeq   EncCnt, LASElectLoop ;
    tax    ;If arrow position changes
    clra   ;
    jsr    LAGotoXY     ;
    lda    #' '         ;
    jsr    LAPrintChar1 ;Clear the old one
    lda    EncCnt       ;
    sta    7, SP        ;
    clra   ;
    ldx    7, SP        ;
    jsr    LAGotoXY     ;
    lda    #Arrow       ;
    jsr    LAPrintChar  ;And draw the new one
    bra    LASElectLoop ;

```

```

LASElected
    lda    7, SP                ;Get Item DoProcPtr
    asla
    add    #$0A
    add    CurMenuPtr+1
    tax
    clra
    adc    CurMenuPtr
    psha
    pulh
    lda    ,X
    bne    LADoItem
    lda    1,X
    beq    LADoNotItem

LADoItem
    lda    ,X
    psha
    ldx    1,X
    pulh
    jsr    ,X

LADoNotItem
    jmp    LAInitMenu

LAEscaped
    jsr    LAPullEncoder        ;Resore old Encoder status
    pula
    ais    #$02
    rts

;*****
;
; Input Device Driver
;
;*****

;Input 2 Digit hex value
;
;On entry
;A=Start value
;On exit
;A=Selected value

LAInputByte
    psha
    lda    CurX
    ldx    CurY
    jsr    LAGotoXY
    lda    1, SP
    jsr    LAPrintHex
    sei

```

Source Code

```

        jsr     LAPushEncoder          ;Save Encoder status
        clr     Min                    ;
        mov     #$FF, Max              ;
        lda     5, SP                  ;
        sta     EncCnt                 ;
        psha                    ;
        lda     #$01                   ;
        sta     EncStep                ;
        cli                    ;
        jsr     LAWaitNotKeyOK         ;
        jsr     LABlinkOn              ;

LASameHex
        jsr     LAKeyOK                ;
        cbeqa   #$01, LASelHex        ;
        lda     EncCnt                 ;
        cbeq    1, SP, LASameHex      ;
        sta     1, SP                 ;
        ldx     CurY                  ;
        lda     CurX                  ;
        deca                    ;
        deca                    ;
        jsr     LAGotoXY               ;
        lda     1, SP                 ;
        jsr     LAPrintHex            ;
        bra     LASameHex             ;

LASelHex
        sei                    ;
        lda     EncCnt                 ;
        sta     6, SP                 ;
        pula                    ;
        jsr     LAPullEncoder          ;
        jsr     LACursOff              ;
        jsr     LAWaitNotKeyOK         ;
        pula                    ;
        rts                          ;

;Wait Input wait for Select or Escape
;
;On exit
;Carry=1 Select, Carry=0 Escape
;

LAWaitInput
        jsr     LAWaitNotKeyOK         ;
        jsr     LAWaitNotPush         ;

WaitInput1
        jsr     LAKeyOK                ;
        sec                    ;
        cbeqa   #$01, WaitInputEnd    ;
        jsr     LAPushOK              ;
        cbeqa   #$00, WaitInput1      ;
        clc                          ;

```

```

WaitInputEnd
    rts                                ;

LAKeyOK
    clra                                ;
    brset    KEY,PORTC,ExitOK          ;If Key not pressed return FALSE
    lda      #Dly100us                 ;
    dbnza    $                          ;Else wait 100us
    brset    KEY,PORTC,ExitOK          ;If noise return FALSE
    inca                                           ;Else return TRUE

ExitOK
    rts                                ;

LAWaitKeyOK
    clr      clrx                       ;

KeyDown
    bsr      LAKeyOK                    ;
    cbeqa    #$00,LAWaitKeyOK          ;
    dbnzx    KeyDown                   ;
    rts                                           ;

LAWaitNotKeyOK
    clr      clrx                       ;

KeyDown1
    bsr      LAKeyOK                    ;
    cbeqa    #$01,LAWaitNotKeyOK      ;
    dbnzx    KeyDown1                  ;
    rts                                           ;

LAWaitNotPush
    clr      clrx                       ;

PushDown
    bsr      LAPushOK                   ;
    cbeqa    #$01,LAWaitNotPush       ;
    dbnzx    PushDown                  ;
    rts                                           ;

LAWaitPush
    clr      clrx                       ;

PushDown1
    bsr      LAPushOK                   ;
    cbeqa    #$00,LAWaitPush          ;
    dbnzx    PushDown1                 ;
    rts                                           ;

LAPushOK
    clra                                ;
    brset    PUSHBUT,PORTD,ExitPush    ;If Key not pressed return FALSE
    lda      #Dly100us                 ;
    dbnza    $                          ;Else wait 100us
    brset    PUSHBUT,PORTD,ExitPush    ;If noise return FALSE
    inca                                           ;Else return TRUE

ExitPush
    rts                                ;

```

Source Code

```
;Save current Encoder status into the stack
;On Entry
; No parameter
;On Exit
; No parameter
```

LAPushEncoder

```
    pulh                    ;
    pulx                    ;
    lda      Min             ;Save Encoder Status
    psha                    ;
    lda      Max             ;
    psha                    ;
    lda      EncCnt          ;
    psha                    ;
    lda      EncStep         ;
    psha                    ;
    jmp      ,X              ;
```

```
;Restore Encoder status from the stack
;On Entry
; No parameter
;On Exit
; No parameter
;Stack status
;
;      SP-1,SP-2=Return address
;      SP-3=EncStep
;      SP-4=EncCnt
;      SP-5=Max
;      SP-6=Min
```

LAPullEncoder

```
    pulh                    ;
    pulx                    ;
    sei                    ;Disable interrupt during
    pula                    ;Encoder status restore
    sta      EncStep         ;
    pula                    ;
    sta      EncCnt          ;
    pula                    ;
    sta      Max             ;
    pula                    ;
    sta      Min             ;
    cli                    ;Enable interrupt
    jmp      ,X              ;
```

```

; IRQ Handling procedures
;
;

LAIRQProc
    psha                ;Save A
    pshx                ;Save X
    ldx      #Dly100us  ;Wait 100uS for anti-debouncing

EncDly
    dbnzx      EncDly   ;
    bih        EndIRQ   ;If IRQ1=1 the interrupt isn't
correct
    lda        EncCnt   ;EncCnt value loaded in Accumulator
    tax
    brclr      ENCB,PORTD,EncUp ;If EncB=1 then increase
    sub        EncStep  ;Otherwise decrease
    sta        EncCnt   ;
    lda        EncStep  ;
    bpl        ChkEncMin ;

ChkEncMax
    cpx        EncCnt   ;
    bhi        SetToMax ;
    lda        EncCnt   ;
    cmp        Max      ;If less than Max is OK
    bls        EndIRQ   ;

SetToMax
    lda        Max      ;
    sta        EncCnt   ;
    bra        EndIRQ   ;

EncUp
    lda        EncStep  ;
    add        EncCnt   ;Increase EncCnt
    sta        EncCnt   ;
    lda        EncStep  ;
    bpl        ChkEncMax ;

ChkEncMin
    cpx        EncCnt   ;
    blo        SetToMin ;
    lda        EncCnt   ;
    cmp        Min      ;
    bhs        EndIRQ   ;If bigger than Min is OK

SetToMin
    lda        Min      ;
    sta        EncCnt   ;

EndIRQ
    bset       2,INTSCR  ;Delete wrong IRQ
    pulx
    pula
    rti

```

Source Code

```

;*****
;
; RS232 Device Driver
;
;*****

LGetRS232
    pshh                ;Save H
    ldhx    #$0400      ;
    sthx    RS232Tmo    ;
    clrh                ;
    tst     SCS1         ;Read Received byte and clear flag
    lda     SCDR         ;
    brclr   EscRec,RS232St,LChkEsc ;Check if $1B already received
    bra     LAEscOk      ;

LChkEsc
    cmp     #$1B         ;If char is Esc=$1B init buffer
    bne     LA232NoCode  ;If not skip
    bset    EscRec,RS232St ;Set flag
    clr     BufferCnt     ;
    bra     LA232NoCode  ;

LEscOk
    ldx     BufferCnt     ;
    sta     RxBuffer,X   ;
    cbeq    #$00,LAInitPack ;If first byte clear CRC
    cpx     RxBuffer     ;
    beq     LChkPack     ;If last byte validate Packet
    bra     LNext232     ;Else store into Rx buffer

LInitPack
    clr     CRC232       ;
    cmp     #LOW(BufferSize) ;If packet too big exit with error
    bhs     LABadPack    ;
    cmp     #$04         ;If packet too small exit with error
    blo     LABadPack    ;

LNext232
    add     CRC232       ;
    sta     CRC232       ;
    inc     BufferCnt     ;
    bra     LA232NoCode  ;

LChkPack
    coma                ;
    cbeq    CRC232,LADecode ;If CRC ok decode command

LABadPack
    lda     #'N'         ;Exit with bad packet message
    clr     RS232St      ;
    clr     BufferCnt     ;
    jsr     LAPutRS232   ;

```

LA232NoCode

```

    pulh                ;
    rti                  ;

```

;Decode Command

LADecode

```

    bclr    EscRec,RS232St    ;
    ldhx    #CmdTab           ;Search command in Bios table
    lda     CmdNum            ;
    bsr     LASrcCmd          ;
    cphx    #$0000            ;If not found
    beq     LASrcUserCmd      ;Search User command
    jsr     ,X                 ;Else execute it
    bra     LA232NoCode       ;

```

LASrcUserCmd

```

    brclr   UserC,SFlags,LABadPack ;
    ldhx    #UserCmdTab         ;Search command in Bios table
    lda     UserCmdNum          ;
    bsr     LASrcCmd            ;
    cphx    #$0000             ;If not found
    beq     LABadPack           ;Error
    jsr     ,X                   ;Else execute it
    bra     LA232NoCode         ;

```

;Search a command in the table pointed by HX

```

;
;On entry
; HX=TablePtr
; A=Num Commands in the table
; Rxbuffer+1->Rxbuffer+3 Command
;On exit
; HX=Command Routine Pointer

```

LASrcCmd

```

    psha                ;Save number of commands
    pshx                ;Save Table addr
    pshh                ;
    clra                ;
    psha                ;Init command counter
    psha                ;Space for char counter

```

NextCmd

```

    clr     1,SP          ;Init char counter

```

NextCmdChar

```

    lda     4,SP          ;
    add     1,SP          ;
    tax     ;
    lda     3,SP          ;
    adc     #$00          ;
    psha    ;

```

Source Code

```

                                pulh                                ;
                                lda                                ,X                                ;Get Cmd char
                                clrh                                ;
                                ldx                                1, SP                                ;
                                inc                                1, SP                                ;
                                cbeqx                            #$03, CmdFound                        ;If 4th char Cmd found
                                cmp                                RxBuffer+1, X                        ;
                                beq                                NextCmdChar                        ;Loop for oll Cmd char
                                inc                                2, SP                                ;If not equal next Cmd
                                lda                                2, SP                                ;
                                cmp                                5, SP                                ;
                                bhs                                CmdTableEnd                        ;Untill last Cmd
                                lda                                #$05                                ;Next Cmd
                                add                                4, SP                                ;
                                sta                                4, SP                                ;
                                clra                                ;
                                adc                                3, SP                                ;
                                sta                                3, SP                                ;
                                bra                                NextCmd                                ;

CmdFound                        lda                                4, SP                                ;
                                tax                                ;
                                lda                                3, SP                                ;
                                psha                                ;
                                pulh                                ;
                                lda                                3, X                                ;
                                psha                                ;
                                ldx                                4, X                                ;
                                pulh                                ;
                                ais                                #$05                                ;
                                rts                                ;

CmdTableEnd                    ldhx                                #$0000                                ;
                                ais                                #$05                                ;
                                rts                                ;

LAPutRS232                     brclr                            7, SCS1, *                                ;
                                sta                                SCDR                                ;
                                rts                                ;

LASendDec                       ldx                                #$0A                                ;
                                clrh                                ;
                                div                                ;
                                pshh                                ;
                                clrh                                ;
                                div                                ;
                                pshh                                ;
                                add                                #$30                                ;

```

```

        bsr      LAPutRS232      ;
        pula     ;
        add      #$30            ;
        bsr      LAPutRS232      ;
        pula     ;
        add      #$30            ;
        bsr      LAPutRS232      ;
        rts                ;

LASEndStr1
        lda      ,X              ;
        beq      LAEndString1    ;
        aix      #$01            ;
        pshh     ;Save pointer   ;
        pshx     ;
        bsr      LAPutRS232      ;
        pulx     ;Restore pointer;
        pulh     ;
        bra      LASEndStr1      ;

LAEndString1
        rts                ;

;Get the length of a string
;
;On entry
; HX=String pointer
;On exit
; A=Len (max 254 char) A=FF length error
; HX=String end pointer

LALen
        pshx     ;Save string pointer
        pshh     ;
        clra     ;

LANextStrLen
        tst      ,X              ;
        beq      LALenEnd        ;
        aix      #$01            ;
        inca     ;
        cbeqa    #$FF, LALenEnd   ;
        bra      LANextStrLen     ;

LALenEnd
        pulh     ;
        pulx     ;
        rts                ;

```

Source Code

```

;*****
;
; Remote commands
;
;*****

;*****
; Programmer Identify Command
;
; Buffer status;
;
; 00      PackLen  (Packet Length 04)
; 01-03   "PID"
; 04      Packet Checksum ($1E)
;*****

CmdPID
        ldhx      #PIDStr
PID1
        jsr       LASendStr1
        rts

;*****
; Erase User Memory Command
;
; Buffer status;
;
; 00      PackLen  (Packet Length 04)
; 01-03   "EUM"
; 04      Packet Checksum ($14)
;*****

CmdEUM
        jsr       LAClr
        ldhx      #EraseFirmMsg
        jsr       LAPrintStr1
        jsr       LAEraseUM
        jmp       LACmdOK

```

```

;*****
; Erase Programming Code Memory Command
;
; Buffer status;
;
; 00      PackLen  (Packet Length 04)
; 01-03   "EPG"
; 04      Packet Checksum ($1F)
;*****

```

CmdEPG

```

        jsr      LAErasePG          ;
        bclr     UserP,SFlags       ;
        jmp      LACmdOK            ;

```

```

;*****
; Erase TX Memory Command
;
; Buffer status;
;
; 00      PackLen  (Packet Length 04)
; 01-03   "ETX"
; 04      Packet Checksum ($0A)
;*****

```

CmdETX

```

        jsr      LAEraseTX          ;
        jmp      LACmdOK            ;

```

```

;*****
; Load TX Memory Command
;
; Buffer status;
;
; 00      PackLen  (Packet Length 04)
; 01-03   "LTX"
; 04      Packet Checksum ($03)
;*****

```

CmdLTX

```

        jsr      LAUpdateTX         ;
        jmp      LACmdOK            ;

```

Source Code

```

;*****
; Load Program Code Memory Command
;
; Buffer status;
;
; 00      PackLen  (Packet Length 04)
; 01-03   "LPG"
; 04      Packet Checksum ($18)
;*****

```

```

CmdLPG
        jsr      LAUpdatePG          ;
LACmdOK
        lda      #'Y'                ;
        clr      RS232St              ;
        clr      BufferCnt             ;
        jsr      LAPutRS232           ;
        rts

```

```

;*****
; Update User Code Command
;
; Buffer status;
;
; 00      PackLen  (Packet Length 04)
; 01-03   "UUC"
; 04      Packet Checksum ($0E)
;*****

```

```

CmdUUC
        jsr      LAClr                ;
        ldhx     #LoadFirmMsg          ;
        jsr      LAPrintStr1           ;
        jsr      LAUpdateUM           ;
        bcc      CmdUUC3              ;
        jsr      LAClr                ;
        ldhx     #ErrFirmMsg           ;
        jsr      LAPrintStr1           ;
        jsr      LAWaitInput           ;
        jmp      LASoftReset           ;
CmdUUC3
        lda      SCDR                 ;
        jmp      LASoftReset           ;

```

```

;*****
;
; Flash programming routine
;
;*****

```

```

;SourcePtr=address of page (128bytes) to be erased

```

```

LAErasePage
    ldhx    SourcePtr
    lda     #%00000010
    sta     FLCR           ;ERASE to 1
    lda     FLBPR
    sta     ,X
    lda     #Dly10us
    dbnza   $              ;Wait 10uS
    lda     #%00001010
    sta     FLCR           ;HVEN to 1
    ldx     #$0A

```

```

ErasedDly
    lda     #Dly100us
    dbnza   $
    dbnzx   ErasedDly
    lda     #%00001000
    sta     FLCR           ;ERASE to 0
    lda     #Dly100us
    dbnza   $
    clra
    sta     FLCR
    rts

```

```

LAErasePageEnd

```

```

LAErasePageLen
    equ     LOW(LAErasePageEnd-LAErasePage)

```

```

;SourcePtr=Start Address
;DestPtr=End Address
;FlashBuffer=Data buffer

```

```

LAProgRange
    ldhx    SourcePtr
    clra
    psha

```

```

NextRow
    lda     #%00000001
    sta     FLCR           ;PGM to 1
    lda     FLBPR
    sta     ,X             ;Dummy write in a flash location
    lda     #Dly10us

```

Source Code

```

    dbnza    $                                ;Wait 10us
    lda      #%00001001                      ;
    sta      FLCR                            ;HVEN to 1
    lda      #Dly5us                         ;
    dbnza    $                                ;Wait 5us

NextRowByte
    ldx      1, SP                          ;
    clrh                                           ;
    lda      FlashBuffer,X                  ;
    ldhx     SourcePtr                      ;
    sta      ,X                             ;
    inc      1, SP                          ;
    aix      #$01                           ;
    sthx     SourcePtr                      ;
    cphx     DestPtr                        ;
    beq      EndRow                         ;
    txa                                           ;
    and      #%00111111                    ;Check if end of row
    beq      EndRow                         ;
    lda      #Dly30us                       ;
    dbnza    $                                ;Wait 30us
    bra      NextRowByte                    ;Next Byte

EndRow
    lda      #Dly30us                       ;
    dbnza    $                                ;Wait 30us
    lda      #%00001000                      ;
    sta      FLCR                            ;PGM to 0
    lda      #Dly10us                       ;
    dbnza    $                                ;Wait 10us
    sta      FLCR                            ;
    cphx     DestPtr                        ;
    bne      NextRow                        ;
    pula                                           ;
    rts                                           ;

LAProgRangeEnd

LAProgRangeLen
    equ      LOW(LAProgRangeEnd-LAProgRange)

LALoadRAM
    psha                                           ;
    clra                                           ;
    psha                                           ;

LALoadRAMLoop
    lda      ,X                             ;
    aix      #$01                           ;
    pshx                                           ;
    pshh                                           ;
    clrh                                           ;
    ldx      3, SP                          ;

```

```

        sta      RxBuffer,X          ;
        pulh                      ;
        pulx                      ;
        inc      1, SP                ;
        dbnz     2, SP, LALoadRAMLoop ;
        pula                      ;
        pula                      ;
        rts                          ;

LAEraseFPage
        ldhx     #LAErasePage        ;
        lda      #LAErasePageLen     ;
        jsr      LALoadRAM            ;Load routine in RAM
        jsr      RxBuffer             ;
        rts                          ;

LAProgFRange
        ldhx     #LAProgRange        ;
        lda      #LAProgRangeLen     ;
        jsr      LALoadRAM            ;Load routine in RAM
        jsr      RxBuffer             ;
        rts                          ;

LALoadUpdate
        ldhx     #$0000              ;

LAUpdateLoop
        lda      LAFirmUpdate,X      ;
        sta      $100,X              ;
        aix      #$01                ;
        cphx     #LAFirmUpdateLen    ;
        bne      LAUpdateLoop        ;
        rts                          ;

LALoadErase
        ldhx     #$0000              ;

LAEraseLoop
        lda      LAEraseUser,X       ;
        sta      $100,X              ;
        aix      #$01                ;
        cphx     #LAEraseUserLen     ;
        bne      LAEraseLoop        ;
        rts                          ;

LAUpdateUM
        bsr      LALoadUpdate        ;
        ldhx     #$D700              ;
        sthx     EndAddr             ;
        ldhx     #$8000              ;
        sthx     FlashTop            ;
        jsr      $100                ;
        rts                          ;

```

Source Code

```

LAEraseUM
    bsr    LALoadErase    ;
    ldhx   #$8000         ;
    sthx   StartAddr     ;
    ldhx   #$D700         ;
    sthx   EndAddr       ;
    jsr    $100           ;
    rts                    ;

LAUpdatePG
    bsr    LALoadUpdate   ;
    ldhx   #$E700         ;
    sthx   EndAddr       ;
    ldhx   #$D700         ;
    sthx   FlashTop      ;
    jsr    $100           ;
    rts                    ;

LAErasePG
    bsr    LALoadErase    ;
    ldhx   #$D700         ;
    sthx   StartAddr     ;
    ldhx   #$E700         ;
    sthx   EndAddr       ;
    jsr    $100           ;
    rts                    ;

LAUpdateTX
    bsr    LALoadUpdate   ;
    ldhx   #$F000         ;
    sthx   EndAddr       ;
    ldhx   #$E700         ;
    sthx   FlashTop      ;
    jsr    $100           ;
    rts                    ;

LAEraseTX
    bsr    LALoadErase    ;
    ldhx   #$E700         ;
    sthx   StartAddr     ;
    ldhx   #$F000         ;
    sthx   EndAddr       ;
    jsr    $100           ;
    rts                    ;

LAEraseUser
    tpa                    ;Save Status register
    psha                   ;
    sei                    ;
    ldhx   StartAddr      ;

```

```

LAEraseUserLoop
    sthx    StartAddr        ;
    lda     #%00000010      ;
    sta     FLCR              ;ERASE to 1
    lda     FLBPR            ;
    sta     ,X               ;
    lda     #Dly10us         ;Wait 10uS
    dbnza   $                ;
    lda     #%00001010      ;
    sta     FLCR              ;HVEN to 1
    ldx     #$0A             ;

LAEraseDly
    lda     #Dly100us        ;
    dbnza   $                ;
    dbnzx   LAEraseDly       ;
    lda     #%00001000      ;
    sta     FLCR              ;ERASE to 0
    lda     #Dly100us        ;
    dbnza   $                ;
    clra    ;
    sta     FLCR              ;
    ldhx    StartAddr        ;
    aix     #$40              ;
    aix     #$40              ;
    cphx    EndAddr          ;
    bne     LAEraseUserLoop  ;
    pula    ;
    tap     ;
    rts     ;

LAEraseUserEnd

LAFirmUpdate
    tpa     ;Save Status register
    psha    ;
    sei     ;
    bsr     LASendYes         ;

LAFirmUpdate1
    bsr     LAGet232          ;
    cbeqa   #'S',LAFirm1     ;
    lda     #'N'              ;
    bsr     LAPut232          ;
    bra     LAFirmUpdate1    ;

LAFirm1
    bsr     LAPut232          ;
    bsr     LAGet232          ;
    psha    ;
    bsr     LAGet232          ;
    tax     ;
    pulh    ;
    bra     LANextL           ;

```

Source Code

```

LAGet232
    brclr    5,SCS1,*
    lda      SCDR
    rts

LAProgErr
    lda      #'N'
    bsr      LAPut232
    pula
    tap
    sec
    rts

LASendYes
    lda      #'Y'

LAPut232
    brclr    7,SCS1,*
    sta      SCDR
    rts

LANextL
    sthx     NumLines
    bsr      LAGet232
    sta      LineLen           ;Save Line length
    sta      ByteCount
    clr      LineSum
    bsr      LAGet232
    sta      StartAddr
    bsr      LAGet232
    sta      StartAddr+1
    ldhx     #$0090

NextLineByte
    bsr      LAGet232
    sta      ,X
    add      LineSum
    sta      LineSum
    incx
    dbnz     LineLen,NextLineByte
    bsr      LAGet232
    cmp      LineSum
    bne      LAProgErr
    ldhx     StartAddr
    lda      #$90
    sta      LASourceData

LANextRow
    lda      #%00000001
    sta      FLCR               ;PGM to 1
    lda      FLBPR
    sta      ,X                 ;Dummy write in a flash location
    lda      #Dly10us           ;Wait 10us
    dbnza    $
    lda      #%00001001
    sta      FLCR               ;HVEN to 1
    lda      Dly5us
    dbnza    $                 ;Wait 10us

```

```

LANextRowByte
    lda    $90                ;
    cphx   EndAddr            ;
    bhs    SkipByte           ;
    cphx   FlashTop           ;
    blo    SkipByte           ;
    sta    ,X                  ;

SkipByte
    lda    #Dly30us           ;
    dbnza  $                  ;Wait 30us
    aix    #$01                ;
    txa                    ;
    and    #%00111111         ;Check if end of row
    beq    LAEndRow           ;
    lda    LASourceData       ;
    inca                    ;
    sta    LASourceData       ;
    dbnz   ByteCount, LANextRowByte ;

LAEndRow
    lda    #%00001000         ;
    sta    FLCR                ;PGM to 0
    lda    #Dly10us           ;Wait 10us
    dbnza  $                  ;
    sta    FLCR                ;
    cbeq   ByteCount, LAEndRow1 ;
    dbnz   ByteCount, LANextRowByte ;

LAEndRow1
    bsr    LASendYes           ;
    ldhx   NumLines            ;
    aix    #$FF                ;
    cphx   #$0000              ;
    bne    LANextL             ;
    pula                    ;
    tap                    ;
    brclr  5, SCS1, *          ;
    lda    SCDR                ;
    clc                    ;
    rts                        ;

LAFirmUpdateEnd

LAEraseUserLen
    equ    LOW(LAEraseUserEnd-LAEraseUser)

LAFirmUpdateLen
    equ    LOW(LAFirmUpdateEnd-LAFirmUpdate)

LASourceData
    equ    LANextRowByte-LAFirmUpdate+$101

```

Source Code

```

;*****
;
; ROM DATA SECTION FOR LOADER AGENT
;
;*****

FirmUpdMsg      DC.B      "Waiting for "
                  DC.B      "Firmware... ", $00
EraseFirmMsg     DC.B      "Erasing      "
                  DC.B      "Firmware... ", $00
LoadFirmMsg      DC.B      "Loading      "
                  DC.B      "Firmware... ", $00
ErrFirmMsg       DC.B      "Error !      "
                  DC.B      "Press a key ", $00

DevName          DC.B      " 68HC908RF2 "
                  DC.B      " Reference "
                  DC.B      " Board    ", $00
AgentMsg         DC.B      "BLIC Agent "
                  DC.B      "Rev 1.0   ", $00
SumMsg           DC.B      "Code :      "
                  DC.B      "Prog :      "
                  DC.B      "Table:      ", $00
FirmOKMsg        DC.B      ":Pass       ", $00
BadFirmMsg       DC.B      ":Fail       ", $00
                  DC.B      "No Firmware ", $00
Contrast         DC.B      $60
VHHVal          DC.B      $FF
PIDStr          DC.B      "RFB A0.0    ", $00

CmdNum           DC.B      $07
CmdTab
                  DC.B      "PID"
                  DC.W      CmdPID
                  DC.B      "EPG"
                  DC.W      CmdEPG
                  DC.B      "ETX"
                  DC.W      CmdETX
                  DC.B      "EUM"
                  DC.W      CmdEUM
                  DC.B      "LPG"
                  DC.W      CmdLPG
                  DC.B      "LTX"
                  DC.W      CmdLTX
                  DC.B      "UUC"
                  DC.W      CmdUUC

CmdTabEnd

```

```

;*****
;
; BIOS Jump Table
;
;*****

IFDEF TestAgent

BiosVectors:    SECTION

WaitInput      jmp     LAWaitInput      ;
KeyOK          jmp     LAKeyOK          ;
WaitKeyOK      jmp     LAWaitKeyOK      ;
WaitNotKeyOK   jmp     LAWaitNotKeyOK   ;
WaitNotPush    jmp     LAWaitNotPush    ;
PushOK         jmp     LAPushOK         ;
WaitPush       jmp     LAWaitPush       ;
PushEncoder    jmp     LAPushEncoder    ;
PullEncoder    jmp     LAPullEncoder    ;
InputByte      jmp     LAInputByte      ;
PowerOff       jmp     LAPowerOff       ;
Dly1ms         jmp     LADly1ms         ;
DlyNms         jmp     LADlyNms         ;
Wait20ms       jmp     LAWait20ms       ;
Wait200ms      jmp     LAWait200ms      ;
Wait1s         jmp     LAWait1s         ;
SetContrast    jmp     LASETContrast    ;
SetVHH         jmp     LASETVHH         ;
WriteDAC       jmp     LAWriteDAC       ;
Wait5uS        jmp     LAWait5uS        ;
Home           jmp     LAHome           ;
CursOn         jmp     LACursOn         ;
CursOff        jmp     LACursOff        ;
BlinkOn        jmp     LABlinkOn        ;
Clr            jmp     LAClr            ;
GotoXY         jmp     LAGotoXY         ;
PrintStr       jmp     LAPrintStr       ;
PrintStr1      jmp     LAPrintStr1      ;
PrintChar      jmp     LAPrintChar      ;
PrintChar1     jmp     LAPrintChar1     ;
PrintHex       jmp     LAPrintHex       ;
PrintShort     jmp     LAPrintShort     ;
PrintDec       jmp     LAPrintDec       ;
InputHex       jmp     LAInputHex       ;
PutRS232       jmp     LAPutRS232       ;
GetRS232       jmp     LAGetRS232       ;
SendDec        jmp     LASendDec        ;
SendStr1       jmp     LASendStr1       ;
EraseFPAGE     jmp     LAEraseFPAGE     ;
ProgFRange     jmp     LAProgFRange     ;
UpdateUM       jmp     LAUpdateUM1     ;
Menu           jmp     LAMenu           ;
SoftReset      jmp     LASoftReset      ;
StrLen         jmp     LAStrLen         ;

ENDIF

```

Source Code

```

;*****
;
; Parameter Flash
;
;*****
Param:          SECTION

NVRegs          DC.B      $D4
                DC.B      $A1
                DC.B      $00

                END

```

A.2.2 UK509 PRM

```

NAMES
ansi.lib
/* other object files to link are passed from the IDF with the linker -Add option
*/
END

SECTIONS
    DIRECT_RAM = READ_WRITE 0x40 TO 0xFF;
    BUFF_RAM   = READ_WRITE 0x0100 TO 0x01FF;
    STACK_RAM  = READ_WRITE 0x0200 TO 0x023F;
    MY_ROM     = READ_ONLY  0x8000 TO 0xEFFF;
    LOADER_ROM = READ_ONLY  0xF000 TO 0xFBFF;
    BIOSVEC_ROM = READ_ONLY  0xFC00 TO 0xFCFF;
    PARAM_ROM  = READ_ONLY  0xFD00 TO 0xFDFF;
END

PLACEMENT
    DEFAULT_ROM, ROM_VAR, STRINGS INTO MY_ROM;
    DEFAULT_RAM                INTO DIRECT_RAM;
    BUFFERS                     INTO BUFF_RAM;
    BiosVectors                 INTO BIOSVEC_ROM;
    Loader                      INTO LOADER_ROM;
    Param                       INTO PARAM_ROM;
    SSTACK                      INTO STACK_RAM;
END

STACKSIZE 0x40

INIT LAInit
VECTOR ADDRESS 0xFFFFE LAInit
VECTOR ADDRESS 0xFFFFA IRQProc
VECTOR ADDRESS 0xFFE4 GetRS232

```

A.3 Transmitter Firmware

```
;Emulated equ 0 ;

;-----
;
; UK804 : Transmitter Sw (inside the MC68HC908RF2)
;-----
;
; SOURCE      : UK804.ASM
; CONTROLLER  : 68HC908RF2
; CLOCK       : 8 Mhz
;
; VERSIONE    : 1.00    28 Feb 2001
;-----
;
;          #$-VIEW:UK804 Schematics-$$

include      "C:\Metrowerks\lib\HC08c\INC\H908RK2.INC"

;          #$-VIEW:RK2 Registers Declarations-$$

;#####
;#
;#          TRANSMISSION FRAME COMPOSITION
;#
;# NNNNNNME-IIIIIIII-IIIIIIII-IIIIIIII-DGGGTTTT-RRRRRRRR
;# | EXTRA |-----ID  NUMBER-----| codice | --CRC--|
;#
;# I=ID NUMBER -> FIX IDENTIFICATION CODE (3 BYTES)
;# G=GROUP     -> SELECTED GROUP
;# D=DIRECTION -> 1=INCREASE 0=DECREASE
;# M=MODE      -> 1=Toggle 0=Normal
;# T=BUTTON    -> BUTTON SELECTED CODE
;# R=CRC       -> CRC
;# E=EXTRA     -> COMMUTATION FLAG (1-> COMMUTATION 0-> IT CHANGES)
;# N=UNUSED    -> THEY MUST BE 0
;#
;#
;#          TEMPORIZATIONS
;#
;# Duration Time of the sync signal = 6ms
;# Duration Time of a Bit           = 2ms
;# Bit 0 -> LowTime=1.33ms HighTime=0.66ms
;# Bit 1 -> LowTime=0.66ms HighTime=1.33ms
;#
;######
```

Source Code

```

; Keyboard Buttons Codification

;      COL0      COL1
;      02 <D--P1--U> 82 ROW0
;      04 <D--P2--U> 84 ROW1
;      06 <D--P3--U> 86 ROW2
;      01 <D--P4--U> 81 ROW3

; Porta A

PLLEn      equ      0      ;Tango PLL Enable
Row0       equ      1      ;Keyboard Row 0
Row1       equ      2      ;Keyboard Row 1
Row2       equ      3      ;Keyboard Row 2
Row3       equ      4      ;Keyboard Row 3
Col0       equ      5      ;Keyboard Column 0
Col1       equ      6      ;Keyboard Column 1
Led        equ      7      ;Led

DDRAInit   equ      %10000001 ;
PortAInit  equ      %00000000 ;

; Porta B

Data       equ      2      ;Data Out to Tango
DClk       equ      3      ;Data Clock from Tango

DDRBInit   equ      %00000100 ;
PortBInit  equ      %00000000 ;

      ORG $80

Buffer0    DS.B      1      ;Control Byte
Buffer1    DS.B      1      ;ID0
Buffer2    DS.B      1      ;ID1
Buffer3    DS.B      1      ;ID2
Buffer4    DS.B      1      ;Key
Buffer5    DS.B      1      ;RC0
Buffer6    DS.B      1      ;RC1
CRC        DS.B      1      ;
BitCount   DS.B      1      ;
LastRow    DS.B      1      ;
LastCol    DS.B      1      ;
TxCnt      DS.B      1      ;

```

; Costanti

```

Dly300ms    equ        $03                ;300ms@8Mhz
MinTx       equ        $03                ;Minimum number of frame transmitted
Framedly    equ        $20                ;Delay between frames
                                                (771*Framedly cycles)

```

codeSec: SECTION

```

XDEF    main
XDEF    KeybIRQ

```

main

```

mov        #%00000011, CONFIG1            ;Diasble COP and enable Stop
                                                instruction

```

```

rsp                                                ;Init stack pointer

```

WarmStart

```

mov        #DDRAInit, DDRA                ;Init PORTA
mov        #PortAInit, PORTA              ;
mov        #DDRBInit, DDRB                ;Init PORTB
mov        #PortBInit, PORTB              ;
bset       PLLEn, PORTA                   ;Start PLL of Tango

```

```

;
; WARNING !!!!! IMASKK is inverted in emulation with MMDS/MMEVS
;

```

ifdef Emulated

```

mov        #%00000011, INTKBSCR            ;Init Keyboard

```

else

```

mov        #%00100001, INTKBSCR            ;

```

endif

```

mov        #%00011110, INTKBIER            ;

```

Sleep

```

bset       ACKK, INTKBSCR                  ;Clear possible Keyboard interrupt
                                                before Stop

```

```

bset       Led, PORTA                      ;Power off led

```

```

bclr       PLLEn, PORTA                   ;Stop PLL of Tango

```

```

;
; WARNING !!!!! Stop instruction does not work with MMDS/MMEVS
;

```

ifdef Emulated

Source Code

```

StopProc
    cli                                ;Emulated Stop instruction
    nop                                ;
    bra    StopProc

else
StopProc
    stop                                ;
    nop                                ;
    bra    StopProc

endif

KeybIRQ
    brclr  7,INTKBSCR,KeyboardProc ;Check if IRQ intetrrupt
    jmp    IRQProc                  ;

KeyboardProc
    bset   2,INTKBSCR                ;Reset Keyboard Flag
    bclr   Led,PORTA                ;Led On
    clrx                                       ;

WaitRow
    lda    PORTA                    ;Read Key Row
    coma                                       ;
    and    #%00011110                ;
    lsra                                       ;
    bne    RowOk                    ;
    dbnza  $                          ;
    dbnzx  WaitRow                  ;If Row=0 wait until timeout
    jmp    ExitKeyb                ;Row timeout

RowOk
    sta    LastRow                  ;Save Row number
    clr    INTKBIER                ;Clear Keyboard interrupt enable

bits
    lda    #%00011110                ;All rows to 1
    ora    DDRA                      ;
    sta    DDRA                      ;
    lda    #%00011110                ;
    ora    PORTA                    ;
    sta    PORTA                    ;
    clrx                                       ;
    mov    #$10,BitCount            ;

ColLoop
    lda    #$40                      ;

KeyLoop
    dbnzx  KeyLoop                  ;
    dbnza  KeyLoop                  ;Wait 49152 cycles

```

```

WaitCol
    lda        PORTA                ;Get Column
    and        #%01100000          ;
    nsa        ;
    lsra       ;
    bne        Col0k                ;
    dbnz       BitCount, ColLoop    ;If Row=0 wait until timeout
    jmp        ExitKeyb            ;Column timeout

Col0k
    sta        LastCol              ;Save column
    lda        #$02                 ;Decode Key
    brset      0, LastRow, SetCol    ;
    lda        #$04                 ;
    brset      1, LastRow, SetCol    ;
    lda        #$06                 ;
    brset      2, LastRow, SetCol    ;
    lda        #$01                 ;

SetCol
    brclr      1, LastCol, SetKey    ;
    ora        #$80                 ;

SetKey
    ;Init Transmission Buffer

    sta        Buffer4              ;Init Key Code
    clr        Buffer0              ;
    lda        ID0                  ;Init ID
    sta        Buffer1              ;
    lda        ID1                  ;
    sta        Buffer2              ;
    lda        ID2                  ;
    sta        Buffer3              ;
    lda        #Dly300ms            ;Wait for the time specified in
                                    ;Accumulator

    psha       ;
    clra       ;
    clrx       ;

ToggledDly
    dbnzx      $                    ;[3*256=768]
    dbnza      ToggledDly          ;[256*771]
    dbnz       1, SP, ToggledDly    ;[197379*Dly300ms]
    pula       ;

    lda        PORTA                ;Check if key still down
    and        #%01100000          ;
    nsa        ;
    lsra       ;
    bne        NoToggle            ;
    bset       0, Buffer0           ;

```

Source Code

NoToggle

```

lda    #$01
add    Buffer6
sta    Buffer6
clra
adc    Buffer5
sta    Buffer5
lda    Buffer6
and    #$0F
beq    NoRotate
tax

```

NextRotate

```

lda    Buffer5
rora
ror    Buffer0
ror    Buffer1
ror    Buffer2
ror    Buffer3
ror    Buffer4
ror    Buffer5
dbnzz NextRotate

```

NoRotate

```

ldhx   #Buffer0
lda    ,X
eor    1,X
sta    1,X
eor    2,X
sta    2,X
eor    3,X
sta    3,X
eor    4,X
sta    4,X
eor    5,X
sta    5,X

```

; Calculate CRC

```

clr    CRC
ldhx   #Buffer6
NextCRCByte
mov     #$08, BitCount
NextCRCBit
lda    0,X
eor    CRC
lsr    CRC
and    #$01
beq    NoXor
lda    #%10001100
eor    CRC
sta    CRC

```

NoXor

```

lda      0,X          ;
rora     ;
ror      0,X          ;
dbnz     BitCount,NextCRCBit ;
decx     ;
cpx      #Buffer0-1   ;
bne      NextCRCByte  ;
lda      CRC          ;
ror      CRC          ;
rora     ;
nsa      ;
sta      CRC          ;

```

```

;#####
;#
;#           Data Timing                      #
;#
;# Timing is derived from Tango DataClock signal (Tango Xtal/64) that is      #
;# 212Khz@13.560Mhz (4.717 us period).                                       #
;# Tango DataClock is used as Timer Module Clock by MCU. When external clock  #
;# is used by Timer Module the prescaler is not available and the clock is    #
;# directly to the timer counter.                                             #
;#
;#####

```

;Transmission Loop

```

mov      #DDRAInit,DDRA      ;
mov      #PortAInit,PORTA    ;
mov      #%00011110,INTKBIER ;
mov      #MinTx,TxCnt        ;
bset     PLLEn,PORTA         ;
mov      #%00110111,TASCR     ;Stop and reset Timer

NextFrame
bset     Data,PORTB          ;
ldx      #Framedly           ;
clra     ;

FrameBlank
clr      TASC0               ;
dbnza    $                   ;
dbnzx    FrameBlank          ;
mov      #%00110111,TASCR     ;Stop and reset Timer
bclr     Data,PORTB          ;
mov      #%00010010,TASC0     ;
ldhx     #$0300               ;Timer modulo=$0300 (3.622ms)
sthx     TAMODH               ;
ldhx     #$0180               ;Compare Value=$0180 (0.603ms)
sthx     TACH0H               ;
bclr     7,TASC0              ;Clear Flag

```

Source Code

```

mov      #%00011110, TASC0      ;Set On Compare and Toggle on
                                   Overflow
bclr     7, TASC0                ;Clear Flag
bclr     5, TASC0                ;Start Timer
brclr    7, TASC0, *             ;Wait
mov      #%00110111, TASC0      ;Stop Timer
ldhx     #$00D2                  ;Timer modulo=$00D2 (990us)
sthx     TACH0H                  ;
clr      TACH0H                  ;
lda      #$46                    ;
sta      TACH0L                  ;Compare Value=1/3 bit time
ldhx     #CRC                    ;HX points to CRC
psha     ;Space in stack for bit counter
mov      #%00011110, TASC0      ;Set On Compare and Toggle on
                                   Overflow
bclr     5, TASC0                ;Start Timer
NextTxByte
lda      #$08                    ;Init Bit Counter
sta      1, SP                  ;
NextTxBit
lda      0, X                    ;
ror      0, X                    ;
ror      0, X                    ;
clr      TACH0H                  ;
lda      #$46                    ;Compare Value=1/3 bit time if bit=1
bcs      Bit1                    ;
lsrla    ;Compare Value=2/3 bit time if bit=0
Bit1
sta      TACH0L                  ;
brclr    7, TASC0, *             ;Wait
bclr     7, TASC0                ;Clear Flag
dbnz     1, SP, NextTxBit        ;
decx     ;
cpx      #Buffer0-1              ;
bne      NextTxByte              ;
mov      #%00110111, TASC0      ;Stop Timer
pula     ;Free stack
clra     ;
dbnza    $                       ;
tst      TxCnt                  ;
beq      ChkKey                  ;
dbnz     TxCnt, NextFrame        ;
ChkKey
lda      PORTA                   ;
coma     ;
and      #%00011110              ;
bne      NextFrame              ;
bclr     PLLEn, PORTA            ;

```



```
ExitKeyb      mov      #DDRAInit, DDRA      ;
               mov      #PortAInit, PORTA    ;
               mov      #%00011110, INTKBIER ;

WaitKeyUp     lda      PORTA                ;
               coma     ;                    ;
               and      #%00011110          ;
               bne      WaitKeyUp           ;
               bset     ACKK, INTKBSCR       ;
               bset     Led, PORTA          ;
               tst      5, SP                ;Test Return Address Low
               bne      DecLow              ;
               dec      4, SP                ;

DecLow        dec      5, SP                ;
               rti      ;                    ;

IRQProc       bset     6, INTKBSCR          ;Reset IRQ Flag
               rti

ID0           DC.B     $54
ID1           DC.B     $23
ID2           DC.B     $F4

               END
```


Appendix B. Schematics

This appendix includes:

- Mother Board schematic — [Figure B-1](#)
- Receiver schematic — [Figure B-2](#)
- Transmitter schematic — [Figure B-3](#)

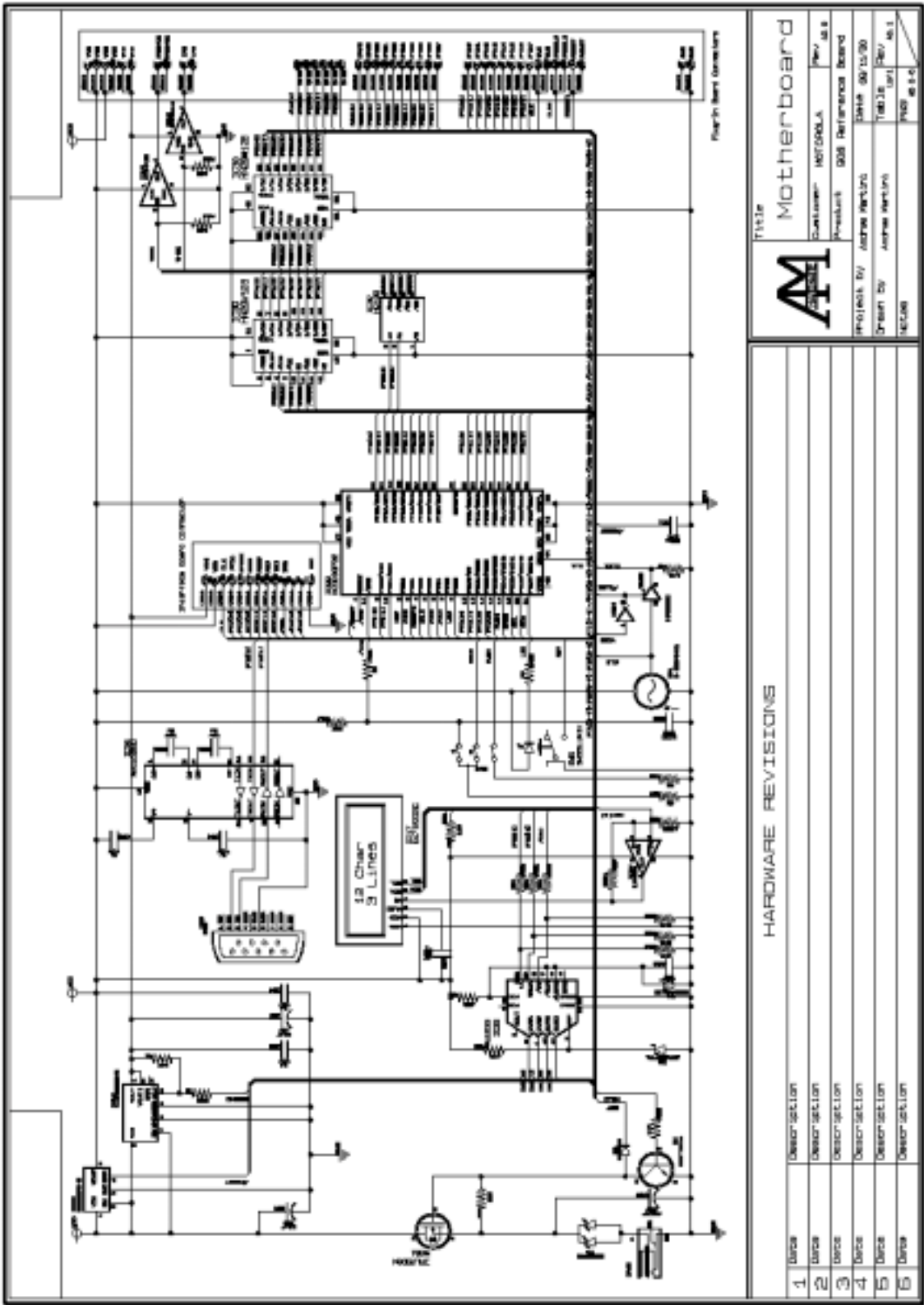


Figure B-1. Mother Board Schematic

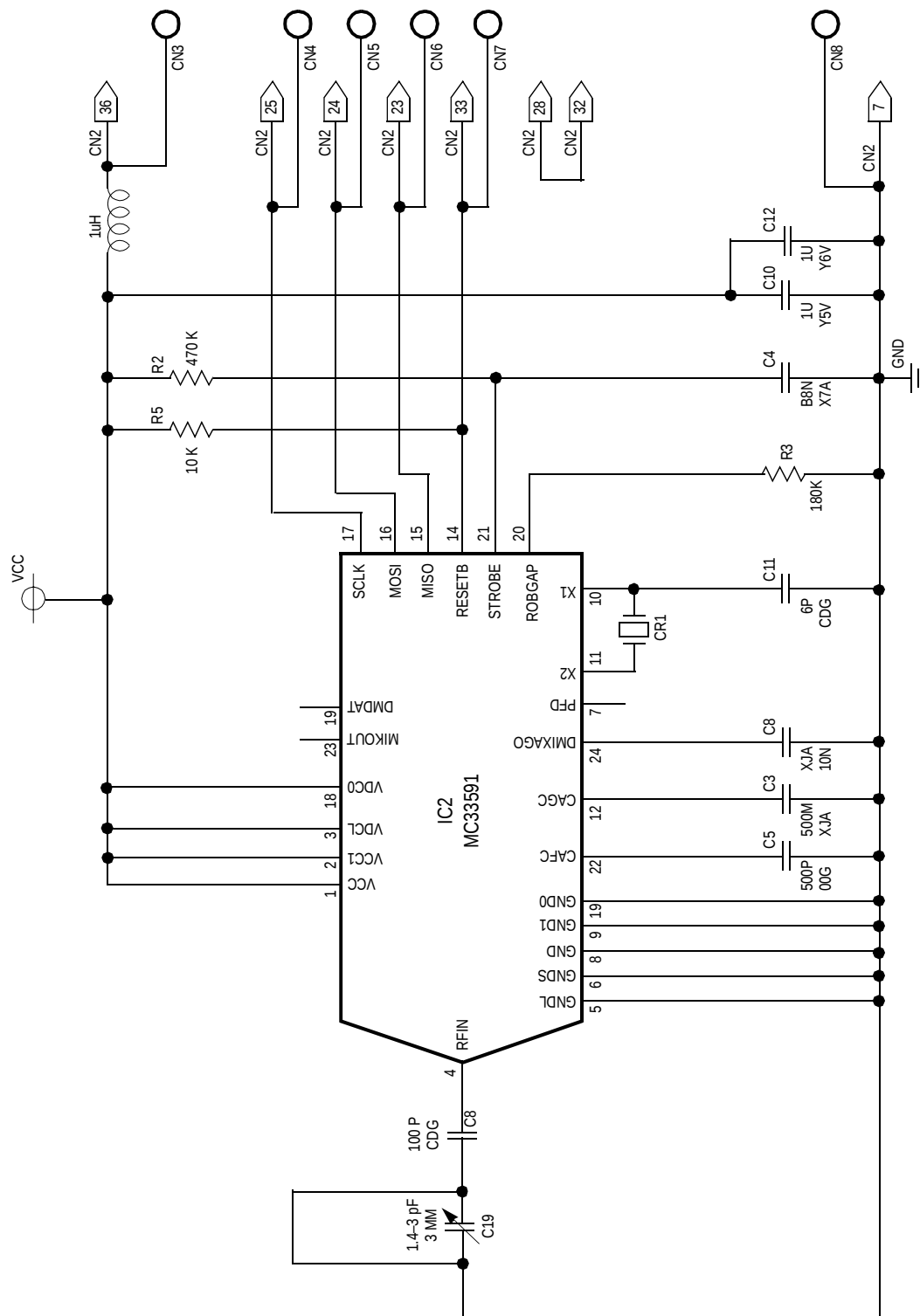


Figure B-2. Receiver Schematic

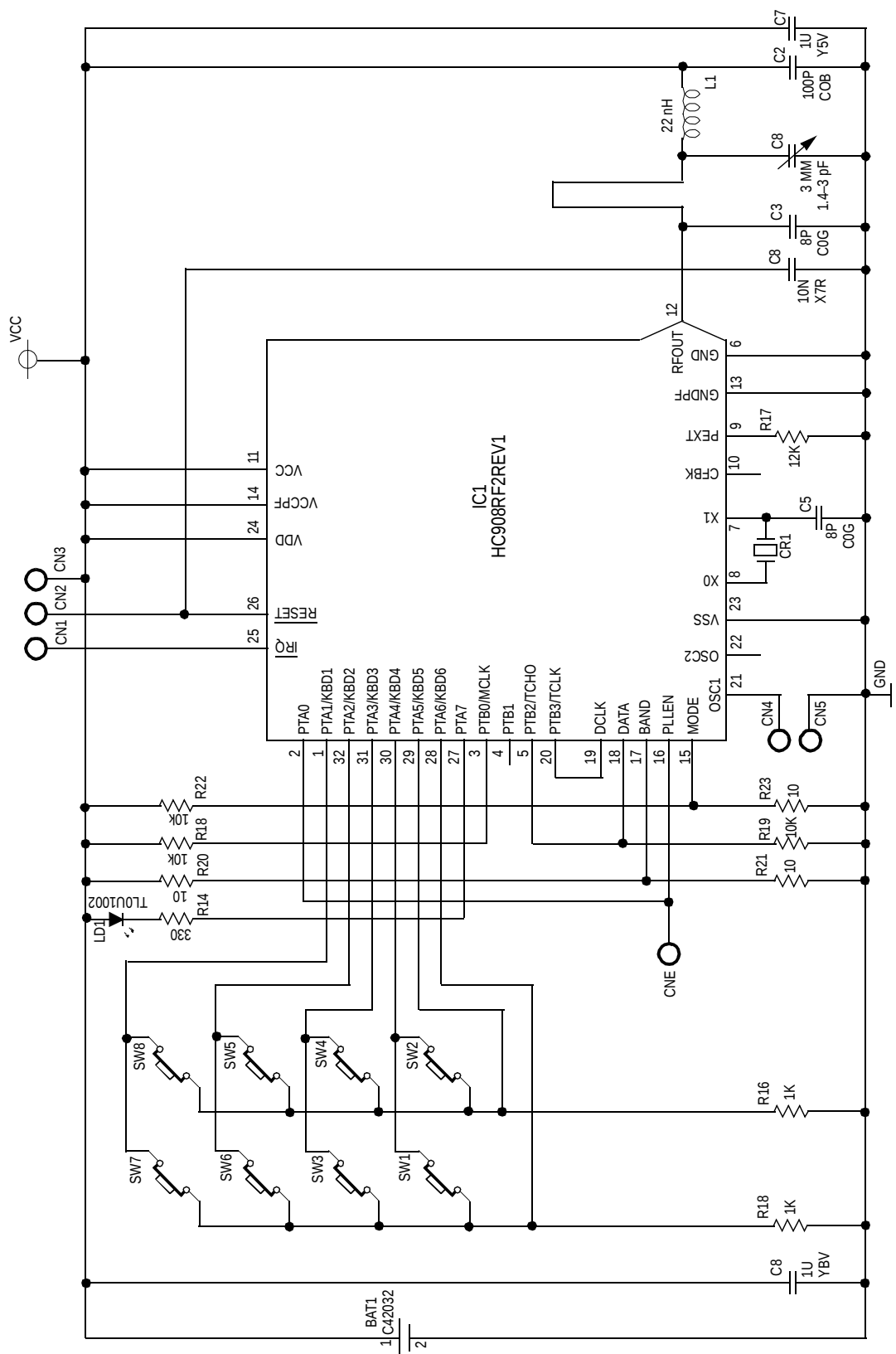


Figure B-3. Transmitter Schematic

Appendix C. Board Layouts

This appendix includes pictorials of the board layouts (top and bottom views) for:

- Transmitter — [Figure C-1](#) and [Figure C-2](#)
- Receiver — [Figure C-3](#) and [Figure C-4](#)
- Mother board — [Figure C-5](#) and [Figure C-6](#)

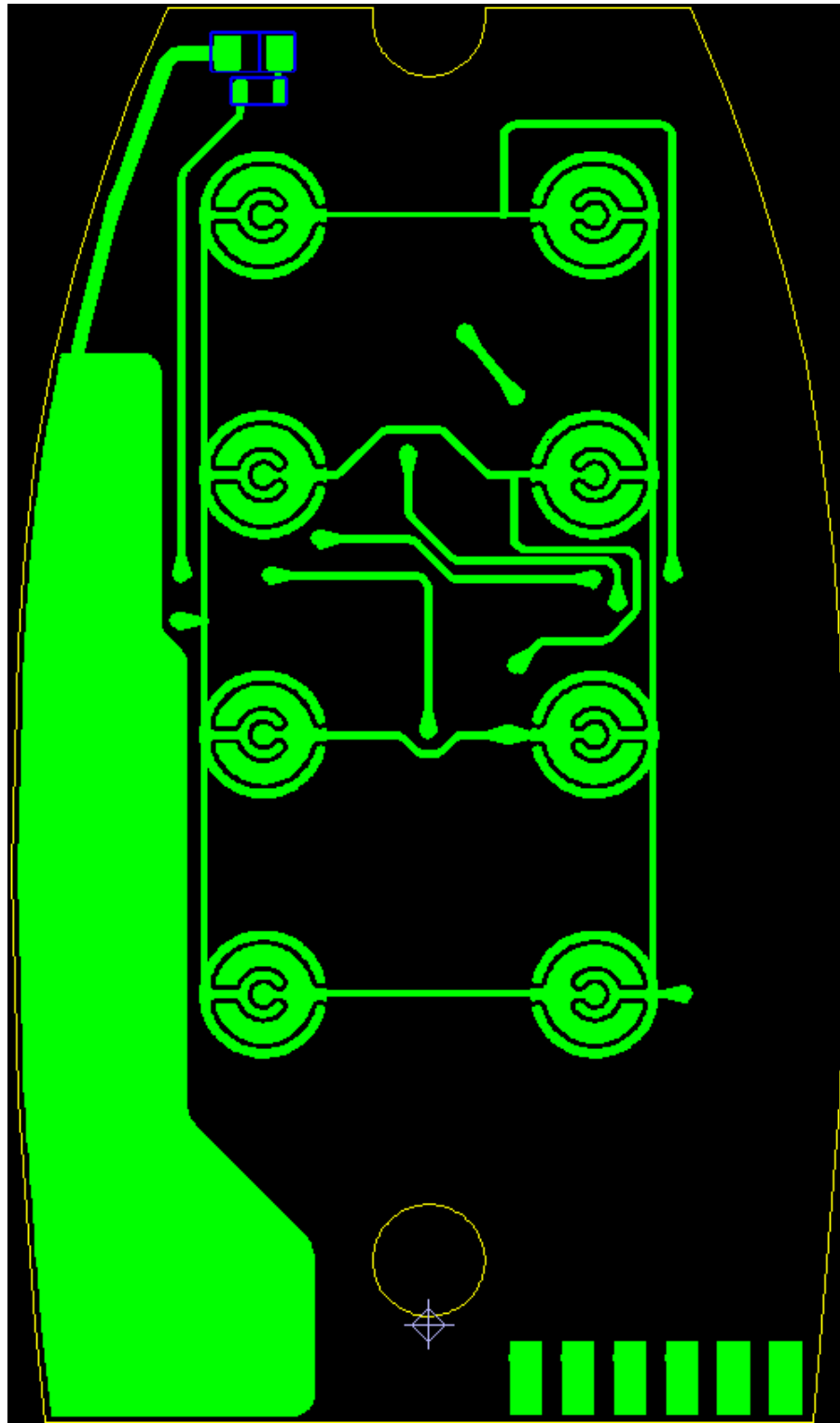


Figure C-1. Transmitter Layout (Top View)

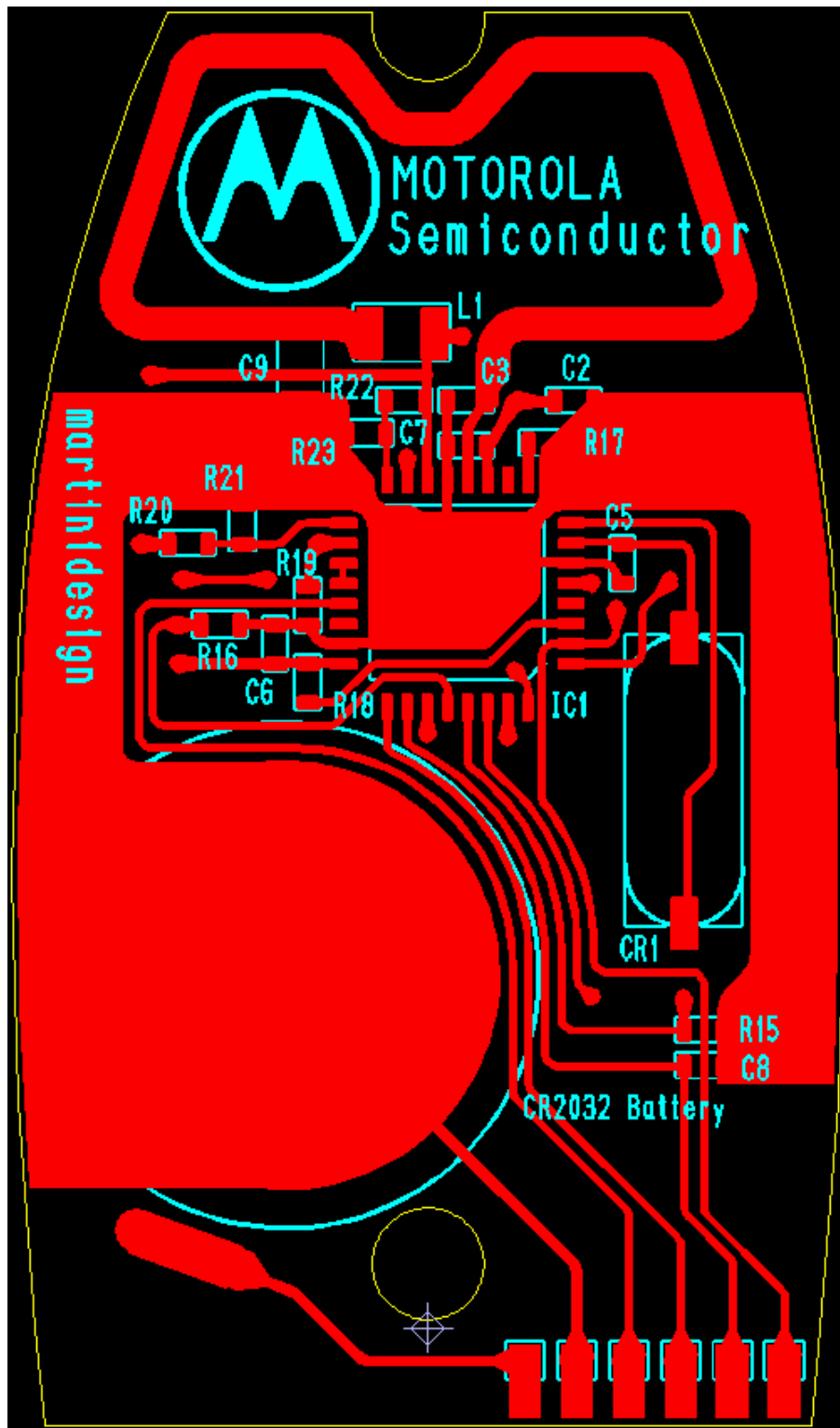


Figure C-2. Transmitter Layout (Bottom View)



Figure C-3. Receiver Layout (Top View)

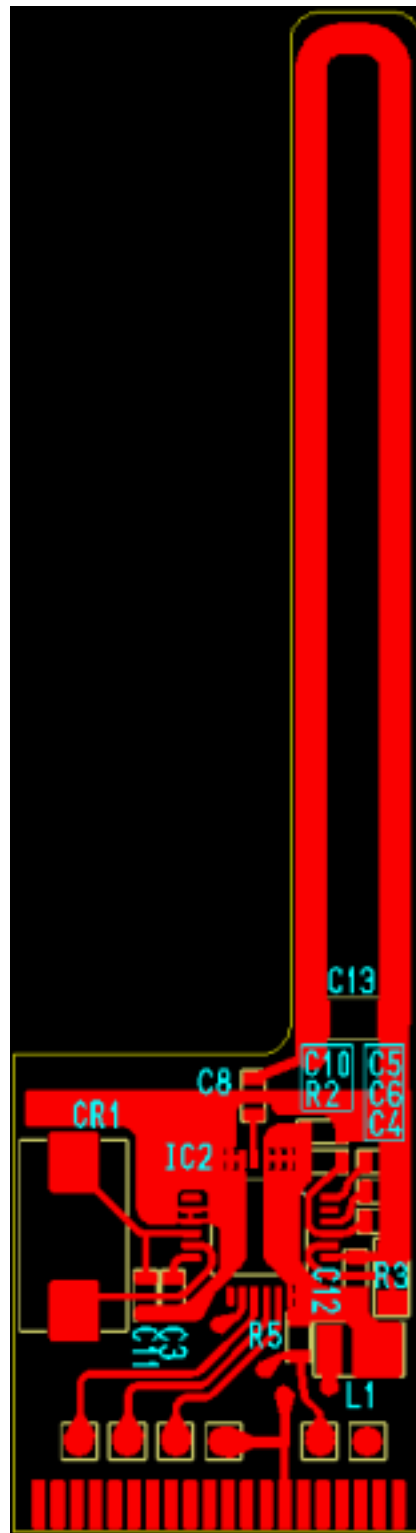


Figure C-4. Receiver Layout (Bottom View)

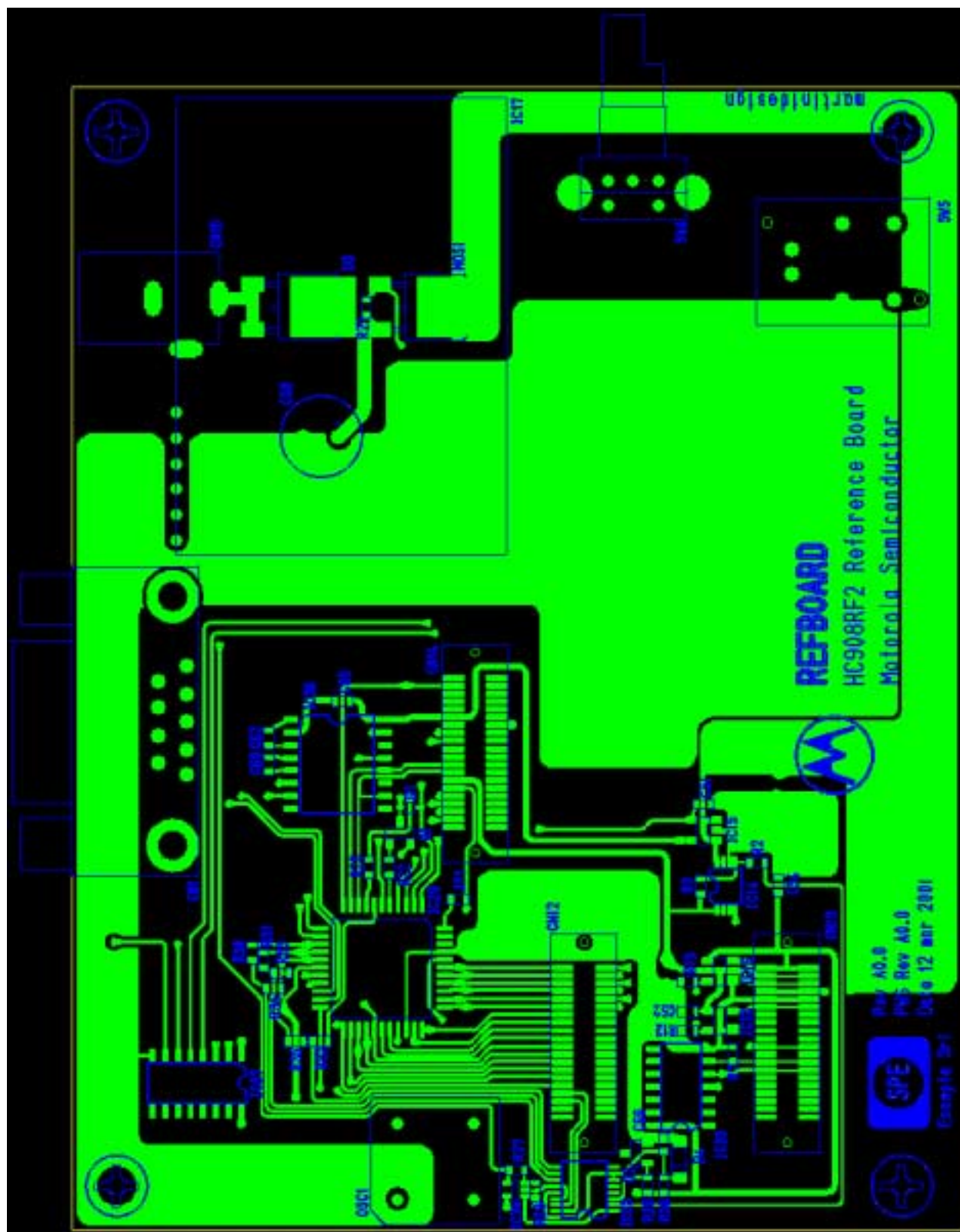


Figure C-5. Mother Board Layout (Top View)

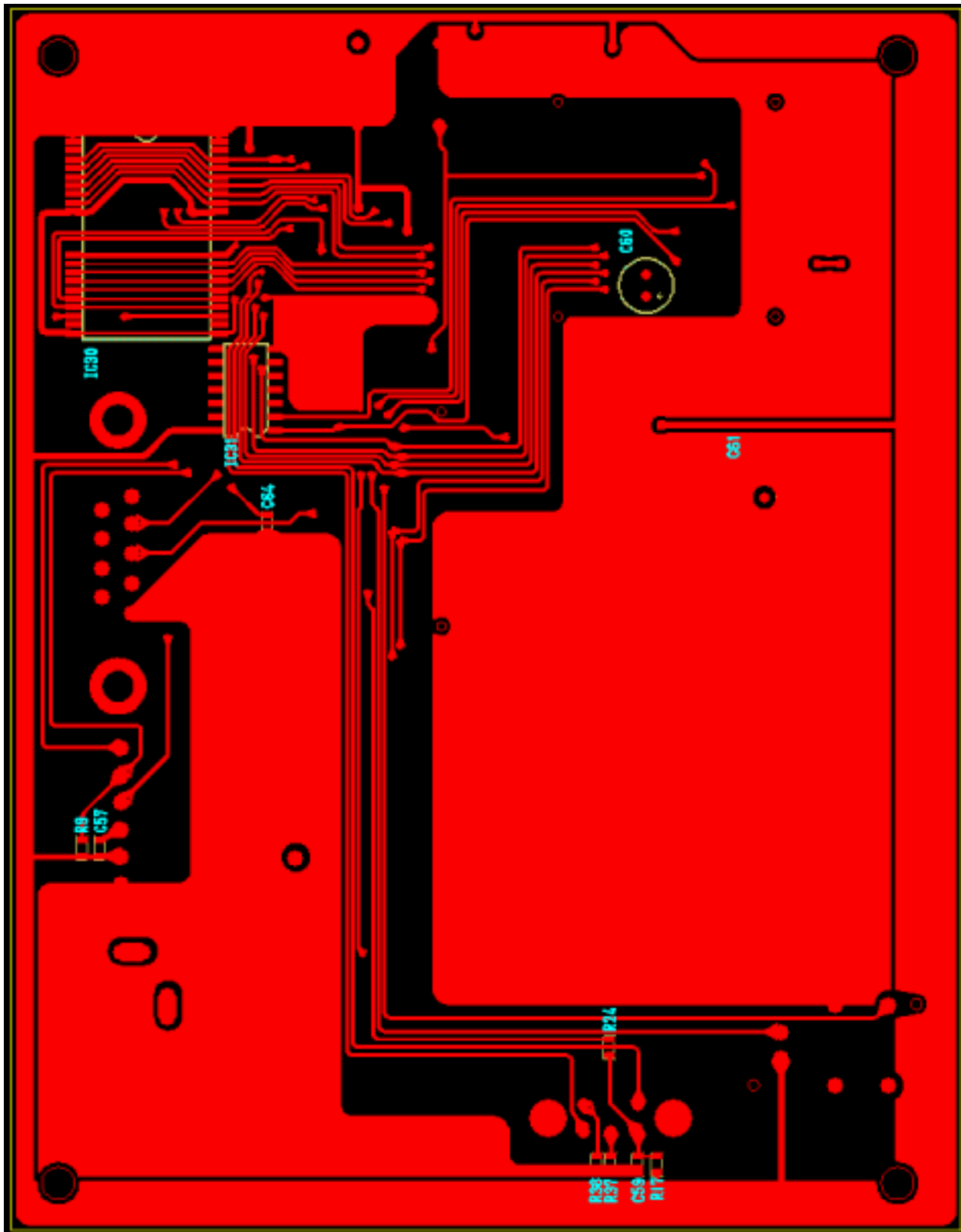


Figure C-6. Mother Board Layout (Bottom View)



Appendix D. Bill of Materials

D.1 Contents

D.2 Introduction147

D.3 Transmitter148

D.4 Receiver149

D.5 Mother Board150

D.2 Introduction

This appendix provides the remote keyless entry (RKE) reference design bill of materials.

Freescale Semiconductor, Inc.



Bill of Materials

D.3 Transmitter

Transmitter PARTLIST

Title TX
Customer Motorola
Author Andrea Martini
Rev A1.0
PWS Rev A0.0
Date 09 Nov 2001

Components

Ref	Value	Package	Function
BAT1		PIH	BATTERY CONTACT
C2	100p	0603	CAPACITOR
C3	8p	0603	CAPACITOR
C5	6p	0603	CAPACITOR
C6	1u	0603	CAPACITOR
C7	1u	0603	CAPACITOR
C8	10n	0603	CAPACITOR
C9	1.4-3pF	SMD	TRIMMER
CR1	13.560MHz	HC49SMX	XTAL
IC1	HC908RF2REV1	LQFP32	MICRO
L1	22nH	008	INDUCTORS
LD1	LED0805	0805	LED
R14	330	0603	RESISTOR
R15	1K	0603	RESISTOR
R16	1K	0603	RESISTOR
R17	12K	0603	RESISTOR
R18	10K	0603	RESISTOR
R19	10K	0603	RESISTOR
R20	10	0603	RESISTOR
R21	10	0603	RESISTOR
R22	10K	0603	RESISTOR
R23	10	0603	RESISTOR
SW			Keyboard



D.4 Receiver

Receiver PARTLIST

Title RX Board
Customer MOTOROLA
Author Andrea Martini
Rev A1.0
PWS Rev A0.0.0
Date 09/11/01

Components

Ref	Value	Package	Function
C10	1u	0603	CAPACITOR
C11	6p	0603	CAPACITOR
C12	1u	0603	CAPACITOR
C13	1.4-3pF	1008	TRIMMER
C3	100n	0603	CAPACITOR
C4	68n	0603	CAPACITOR
C5	100p	0603	CAPACITOR
C6	10n	0603	CAPACITOR
C8	100p	0603	CAPACITOR
CR1	13.5806MHz	SMD3118	XTAL
IC2	MC33591	LQFP24	IC
L1	1uH	1008	INDUCTORS
R2	470K	0603	RESISTOR
R3	180K	0805	RESISTOR
R5	10K	0603	RESISTOR

Bill of Materials

D.5 Mother Board

MotherBoard PARTLIST

Title Motherboard
Customer MOTOROLA
Author Andrea Martini
Rev A0.0
PWS Rev A0.0.0
Date 09/11/00

Components

Ref	Value	Package	Function
C34	1u	0603	CAPACITOR
C51	100n	0603	CAPACITOR
C52	1u	0603	CAPACITOR
C57	100n	0603	CAPACITOR
C58	470uF	8/16-VERT	CAPACITOR
C59	100n	0603	CAPACITOR
C60	10uF	5/11-VERT	CAPACITOR
C61	10uF	5/11-VERT	CAPACITOR
C62	1u	0603	CAPACITOR
C63	1u	0603	CAPACITOR
C64	1u	0603	CAPACITOR
C65	1u	0603	CAPACITOR
C66	1u	0603	CAPACITOR
CN12	FH40		CONNECTOR
CN13	FH40		CONNECTOR
CN14	FH40		CONNECTOR
CN15	PLUG2.5mm		CONNECTOR
CN29	DB9		CONNECTOR
CR1	9.8304MHZ	DIP8	OSC
D1	1SS355	SOD323	DIODE
D2	ZRA250F01	SOT23	REFERENCE
D3	MBRD835	DPACK	DIODE
D4	MMSZ5225B	SOD123	ZENER
IC13	MIC4416	SOT143	MOSFET
IC14	MIC5206BMM	MSOP8	REGULATOR
IC15	MIC5206	SOT23-5	REGULATOR
IC16	MIC4416	SOT143	MOSFET

—Continued on next page —

Ref	Value	Package	Function
IC17	EA7123I2C		LCD
IC29	HC908GP32	QFP44	MCU
IC30	HN29W128	48TSOP	Memory
IC31	74HC04D	SO14	LOGIC
IC33	MAX533BCEE	SSOP16	DAC
IC34	LMV321	SC70	OPERATIONAL
IC35	74HC125D	SO14	LOGIC
IC36	MAX232SO	SO16W	INTERFACE
IC37	74HC139D	SO16	LOGIC
MOS1	IRL9024	D-PACK	MOSFET
Q1	SMBT3904	SOT23	BJT
R12	10K	0603	RESISTOR
R13	10K	0603	RESISTOR
R17	10K	0603	RESISTOR
R2	56K	0603	RESISTOR
R21	10K	0603	RESISTOR
R22	10K	0603	RESISTOR
R24	1K	0603	RESISTOR
R25	1K	0603	RESISTOR
R26	10K	0603	RESISTOR
R27	100K	0603	RESISTOR
R28	100K	0603	RESISTOR
R3	10K	0603	RESISTOR
R30	10K	0603	RESISTOR
R31	10K	0603	RESISTOR
R32	10K	0603	RESISTOR
R33	10K	0603	RESISTOR
R34	10K	0603	RESISTOR
R35	10K	0603	RESISTOR
R36	470	0603	RESISTOR
R37	1K	0603	RESISTOR
R38	1K	0603	RESISTOR
R39	1K	0603	RESISTOR
R7	10K	0603	RESISTOR
R9	10K	0603	RESISTOR
SW4	EC11B15242	Oriz	SWITCHES
SW5	SW5511M1X	PIH	SWITCHES





Freescale Semiconductor, Inc.

Freescale Semiconductor, Inc.

**For More Information On This Product,
Go to: www.freescale.com**

HOW TO REACH US:**USA/EUROPE/LOCATIONS NOT LISTED:**

Motorola Literature Distribution;
P.O. Box 5405, Denver, Colorado 80217
1-303-675-2140 or 1-800-441-2447

JAPAN:

Motorola Japan Ltd.; SPS, Technical Information Center,
3-20-1, Minami-Azabu Minato-ku, Tokyo 106-8573 Japan
81-3-3440-3569

ASIA/PACIFIC:

Motorola Semiconductors H.K. Ltd.;
Silicon Harbour Centre, 2 Dai King Street,
Tai Po Industrial Estate, Tai Po, N.T., Hong Kong
852-26668334

TECHNICAL INFORMATION CENTER:

1-800-521-6274

HOME PAGE:

<http://www.motorola.com/semiconductors>

Information in this document is provided solely to enable system and software implementers to use Motorola products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Motorola data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part.



Motorola and the Stylized M Logo are registered in the U.S. Patent and Trademark Office. digital dna is a trademark of Motorola, Inc. All other product or service names are the property of their respective owners. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

© Motorola, Inc. 2002

DRM005/D

**For More Information On This Product,
Go to: www.freescale.com**