microID™ 13.56 MHz RFID System Design Guide

INCLUDES:

- **Passive RFID Basics Application Note**
- **MCRF355/360 Data Sheet**
- **Microchip Development Kit Sample Format**
- **MCRF355/360 Factory Programming Support (SQTP^{SM})**
- **MCRF355/360 Applications Application Note**
- **Antenna Circuit Design Application Note**
- **13.56 MHz Reader Reference Design**
- **Contact Programmer Reference Design**

NEW CUSTOMER NOTIFICATION SYSTEM

Register on our web site (www.microchip.com/cn) to receive the most current information on our products.

DATA SHEET MARKINGS

Microchip uses various data sheet markings to designate each document phase as it relates to the product development stage. The markings appear at the bottom of the data sheet, between the copyright and document and page numbers. The definitions for each marking are provided below for your use.

Marking	Description
Advance Information	The information is on products in the design phase. Your designs should not be finalized with this information as revised information will be published when the product becomes available.
Preliminary	This is preliminary information on new products in production but not yet fully characterized. The specifications in these data sheets are subject to change without notice. Before you finalize your design, please ensure that you have the most current revision of the data sheet by contacting your Microchip sales office.
No Marking	Information contained in the data sheet is on products in full production.

Table of Contents

Table of Contents

AN680

Passive RFID Basics

Author: *Pete Sorrells*
Microchip Technology Inc.

INTRODUCTION

Radio Frequency Identification (RFID) systems use radio frequency to identify, locate and track people, assets, and animals. Passive RFID systems are composed of three components – an interrogator (reader), a passive tag, and a host computer. The tag is composed of an antenna coil and a silicon chip that includes basic modulation circuitry and non-volatile memory. The tag is energized by a time-varying electromagnetic radio frequency (RF) wave that is transmitted by the reader. This RF signal is called a *carrier signal*. When the RF field passes through an antenna coil, there is an AC voltage generated across the coil. This voltage is rectified to supply power to the tag. The information stored in the tag is transmitted back to the reader. This is often called backscattering. By detecting the backscattering signal, the information stored in the tag can be fully identified.

DEFINITIONS

Reader

Usually a microcontroller-based unit with a wound output coil, peak detector hardware, comparators, and firmware designed to transmit energy to a tag and read information back from it by detecting the backscatter modulation.

Tag

An RFID device incorporating a silicon memory chip (usually with on-board rectification bridge and other RF front-end devices), a wound or printed input/output coil, and (at lower frequencies) a tuning capacitor.

Carrier

A Radio Frequency (RF) sine wave generated by the reader to transmit energy to the tag and retrieve data from the tag. In these examples the ISO frequencies of 125 kHz and 13.56 MHz are assumed; higher frequencies are used for RFID tagging, but the communication methods are somewhat different. 2.45 GHz, for example, uses a true RF link. 125 kHz and 13.56 MHz, utilize transformer-type electromagnetic coupling.

Modulation

Periodic fluctuations in the amplitude of the carrier used to transmit data back from the tag to the reader.

Systems incorporating passive RFID tags operate in ways that may seem unusual to anyone who already understands RF or microwave systems. There is only one transmitter – the passive tag is not a transmitter or transponder in the purest definition of the term, yet bidirectional communication is taking place. The RF field generated by a tag reader (the energy transmitter) has three purposes:

1. **Induce enough power into the tag coil to energize the tag.** Passive tags have no battery or other power source; they must derive all power for operation from the reader field. 125 kHz and 13.56 MHz tag designs must operate over a vast dynamic range of carrier input, from the very near field (in the range of 200 V_{PP}) to the maximum read distance (in the range of 5 V_{PP}).
2. **Provide a synchronized clock source to the tag.** Many RFID tags divide the carrier frequency down to generate an on-board clock for state machines, counters, etc., and to derive the data transmission bit rate for data returned to the reader. Some tags, however, employ on-board oscillators for clock generation.
3. **Act as a carrier for return data from the tag.** Backscatter modulation requires the reader to peak-detect the tag's modulation of the reader's own carrier. See page 2 for additional information on backscatter modulation.

SYSTEM HANDSHAKE

Typical handshake of a tag and reader is as follows:

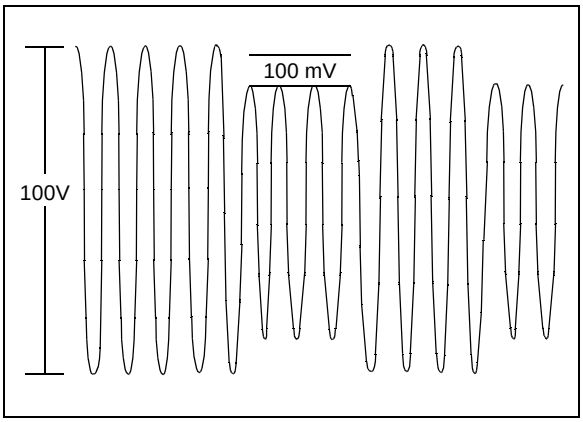
1. The reader continuously generates an RF carrier sine wave, watching always for modulation to occur. Detected modulation of the field would indicate the presence of a tag.
2. A tag enters the RF field generated by the reader. Once the tag has received sufficient energy to operate correctly, it divides down the carrier and begins clocking its data to an output transistor, which is normally connected across the coil inputs.
3. The tag's output transistor shunts the coil, sequentially corresponding to the data which is being clocked out of the memory array.
4. Shunting the coil causes a momentary fluctuation (dampening) of the carrier wave, which is seen as a slight change in amplitude of the carrier.
5. The reader peak-detects the amplitude-modulated data and processes the resulting bitstream according to the encoding and data modulation methods used.

BACKSCATTER MODULATION

This terminology refers to the communication method used by a passive RFID tag to send data back to the reader. By repeatedly shunting the tag coil through a transistor, the tag can cause slight fluctuations in the reader's RF carrier amplitude. The RF link behaves essentially as a transformer; as the secondary winding (tag coil) is momentarily shunted, the primary winding (reader coil) experiences a momentary voltage drop. The reader must peak-detect this data at about 60 dB down (about 100 mV riding on a 100V sine wave) as shown in Figure 1.

This amplitude-modulation loading of the reader's transmitted field provides a communication path back to the reader. The data bits can then be encoded or further modulated in a number of ways.

FIGURE 1: AMPLITUDE – MODULATED BACKSCATTERING SIGNAL

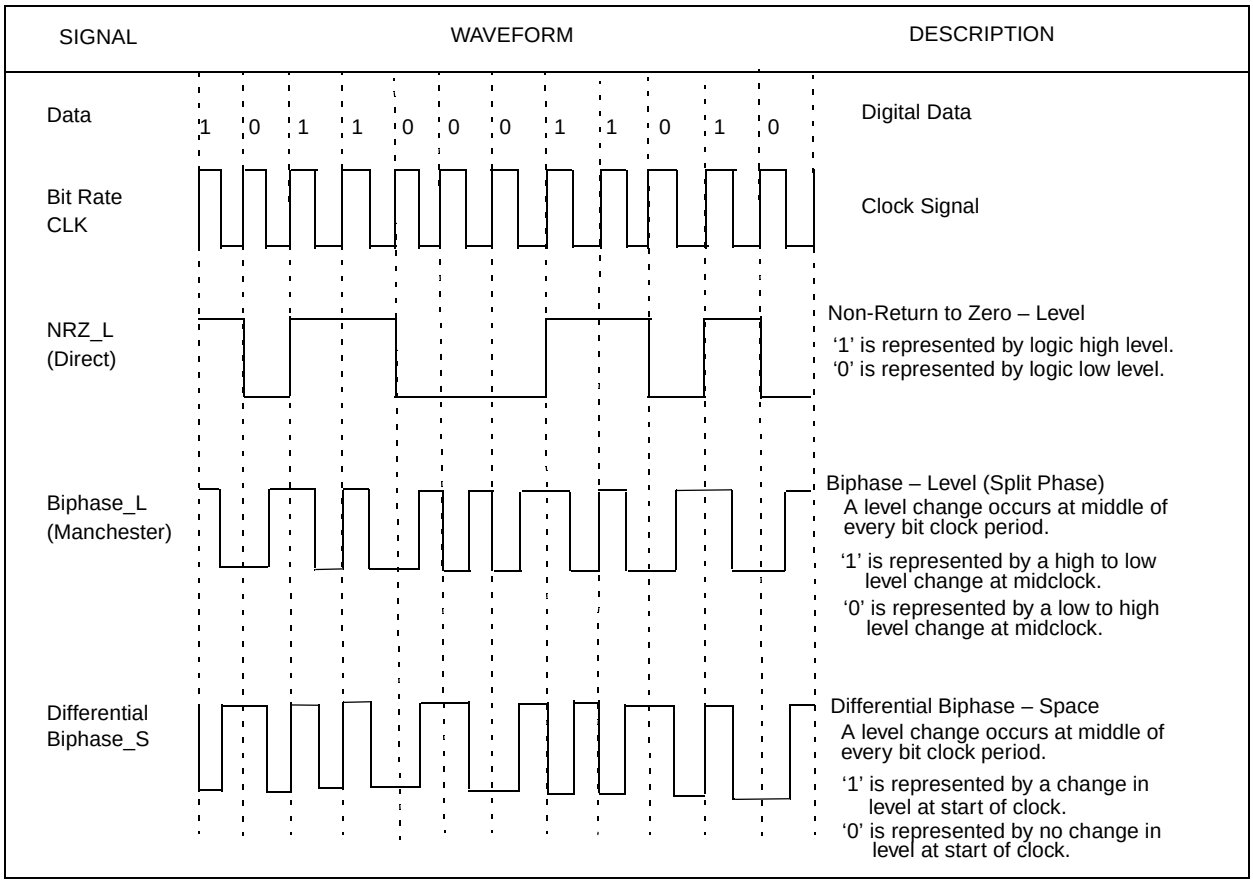


DATA ENCODING

Data encoding refers to processing or altering the data bitstream in-between the time it is retrieved from the RFID chip's data array and its transmission back to the reader. The various encoding algorithms affect error recovery, cost of implementation, bandwidth, synchronization capability, and other aspects of the system design. Entire textbooks are written on the subject, but there are several popular methods used in RFID tagging today:

1. **NRZ (Non-Return to Zero) Direct.** In this method no data encoding is done at all; the 1's and 0's are clocked from the data array directly to the output transistor. A low in the peak-detected modulation is a '0' and a high is a '1'.
2. **Differential Biphase.** Several different forms of differential biphase are used, but in general the bitstream being clocked out of the data array is modified so that a transition always occurs on every clock edge, and 1's and 0's are distinguished by the transitions within the middle of the clock period. This method is used to embed clocking information to help synchronize the reader to the bitstream; and because it always has a transition at a clock edge, it inherently provides some error correction capability. Any clock edge that does not contain a transition in the data stream is in error and can be used to reconstruct the data.
3. **Biphase_L (Manchester).** This is a variation of biphase encoding in which there is not always a transition at the clock edge.

FIGURE 2: VARIOUS DATA CODING WAVEFORMS



DATA MODULATION

Although all the data is transferred to the host by amplitude-modulating the carrier (backscatter modulation), the actual modulation of 1's and 0's is accomplished with three additional modulation methods:

1. **Direct.** In direct modulation, the Amplitude Modulation of the backscatter approach is the only modulation used. A high in the envelope is a '1' and a low is a '0'. Direct modulation can provide a high data rate but low noise immunity.
2. **FSK (Frequency Shift Keying).** This form of modulation uses two different frequencies for data transfer; the most common FSK mode is Fc/8/10. In other words, a '0' is transmitted as an amplitude-modulated clock cycle with period corresponding to the carrier frequency divided by 8, and a '1' is transmitted as an amplitude-modulated clock cycle period corresponding to the carrier frequency divided by 10. The amplitude modulation of the carrier thus switches from Fc/8 to Fc/10 corresponding to 0's and 1's in the bitstream, and the reader has only to count cycles between the peak-detected clock edges to decode the data. FSK allows for a simple reader design, provides very strong noise immunity, but suffers from a lower data rate than some other forms of data modulation. In Figure 3, FSK data modulation is used with NRZ encoding.
3. **PSK (Phase Shift Keying).** This method of data modulation is similar to FSK, except only one frequency is used, and the shift between 1's and 0's is accomplished by shifting the phase of the backscatter clock by 180 degrees. Two common types of PSK are:
   - Change phase at any '0', or
   - Change phase at any data change (0 to 1 or 1 to 0).

   PSK provides fairly good noise immunity, a moderately simple reader design, and a faster data rate than FSK. Typical applications utilize a backscatter clock of Fc/2, as shown in Figure 4.

FIGURE 3: FSK MODULATED SIGNAL, FC/8 = 0, FC/10 = 1

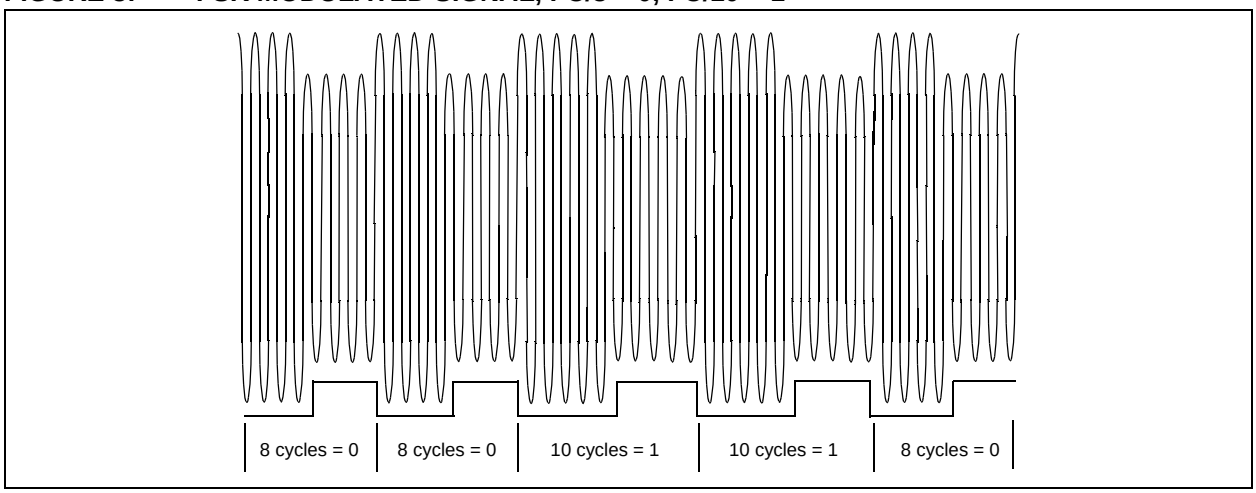
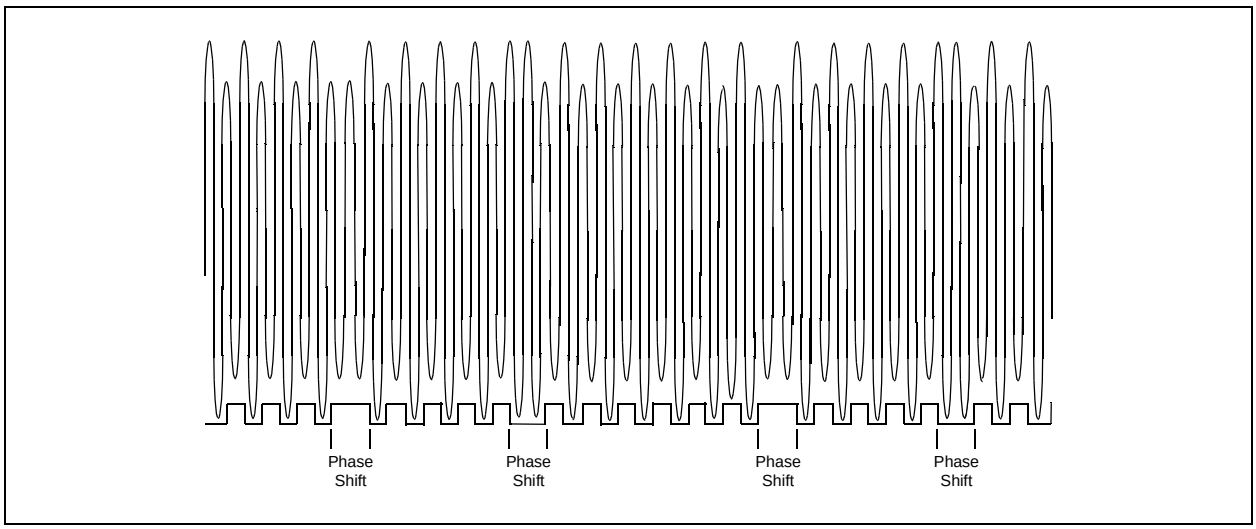


FIGURE 4: PSK MODULATED SIGNAL



ANTICOLLISION

In many existing applications, a single-read RFID tag is sufficient and even necessary: animal tagging and access control are examples. However, in a growing number of new applications, the simultaneous reading of several tags in the same RF field is absolutely critical: library books, airline baggage, garment, and retail applications are a few.

In order to read multiple tags simultaneously, the tag and reader must be designed to detect the condition that more than one tag is active. Otherwise, the tags will all backscatter the carrier at the same time, and the amplitude-modulated waveforms shown in Figures 3 and 4 would be garbled. This is referred to as a *collision*. No data would be transferred to the reader. The tag/reader interface is similar to a serial bus, even though the "bus" travels through the air. In a wired serial bus application, arbitration is necessary to prevent bus contention. The RFID interface also requires arbitration so that only one tag transmits data over the "bus" at one time.

A number of different methods are in use and in development today for preventing collisions; most are patented or patent pending, but all are related to making sure that only one tag "talks" (backscatters) at any one time. See the *MCRF355/360 Data Sheet* (page 7) and the *13.56 MHz Reader Reference Design* (page 47) chapters for more information regarding the MCRF355/360 anticollision protocol.

NOTES:

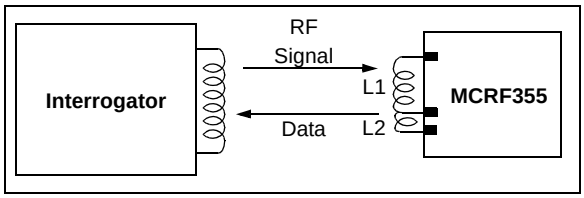
MCRF355/360

13.56 MHz Passive RFID Device with Anticollision

FEATURES

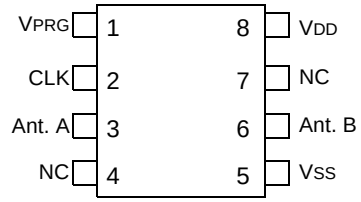
- Frequency of operation: 13.56 MHz
- Built-in anticollision algorithm for reading up to 50 tags in the same RF field
- "Cloaking" feature minimizes the detuning effects of adjacent tags
- Manchester coding protocol
- Data modulation frequency: 70 kHz
- 154 bits of user-programmable memory
- Contact programming or factory-programmed options
- Very low power CMOS design
- Die, wafer, PDIP or SOIC package options
- On-chip 100 pF resonance capacitor (MCRF360)
- Read-only device after programming

APPLICATION



PACKAGE TYPE

PDIP/SOIC



DESCRIPTION

The MCRF355 and MCRF360 are Microchip's newest additions to the microID™ family of RFID tagging devices. They are uniquely designed read-only passive Radio Frequency Identification (RFID) devices with an advanced anticollision feature, operating at 13.56 MHz. The device is powered remotely by rectifying RF magnetic fields that are transmitted from an interrogator (reader).

The device has a total of six pads (see Die Layout). Three are used to connect the external resonant circuit elements. The additional three pads are used for programming and testing of the device.

The device needs two external antenna coils (L1 and L2) to pick up the RF magnetic fields and also to send back encoded (modulated) data to the reader. The two antenna coils are connected in series. The first coil (L1) is connected between Antenna Pad A and Antenna Pad B. The second coil (L2) is connected between Antenna Pad B and VSS. The MCRF355 requires an external capacitor to form a resonant circuit along with the antenna coils. See Figure 6-2 for details.

The MCRF360 has 100 pF of internal resonance capacitor between Antenna Pad A and VSS (across the coils). This capacitance can be utilized to form a tuned LC circuit along with the external antenna coils. See Section 6.2 for external resonant circuits.

The device includes a modulation transistor that is located between Antenna Pad B and VSS. This modulation gate is used to send data to the reader. The modulation transistor is designed to result in approximately 3Ω of resistance between Drain, which is connected to Antenna Pad B, and Source, which is connected to VSS, when it is turned-on.

The LC circuit is tuned to the operating frequency (13.56 MHz) of the reader when the modulation transistor is in a turned-off condition. This condition is called *uncloaking*.

As the modulation transistor turns on, there will be a shorting effect across L2 due to the 3Ω resistance across it. This results in a change of the inductance of the antenna coil, and, therefore, the circuit no longer resonates at 13.56 MHz. This condition is called *cloaking*.

The occurrence of the cloaking and uncloaking of the device is controlled by the modulation signal that turns the modulation transistor on and off, resulting in communication from the device to the reader.

The data stream consists of 154 bits of Manchester-encoded data. The code waveforms are shown in Figure 6-3. The data is sent to the reader by modulating (AM) the carrier signal (13.56 MHz). After completion of the data transmission, the device goes into sleep mode for 100 ms ± 40%. The device repeats the transmitting and sleep cycles as long as it is energized.

Sleep time is determined by a built-in, low-current timer. The variation of sleep time is approximately ± 20%. The variation of sleep time between each device results in a randomness of the time slot. Each device wakes up and transmits its data in a different

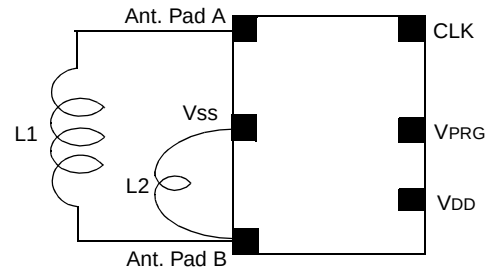
microID is a trademark of Microchip Technology Inc.

time slot with respect to each other. Based on this scenario, the reader is able to read many tags that are in the same RF field.

The device has a total of 154 bits of contact reprogrammable memory. All bits are reprogrammable by a contact programmer. A contact programmer (part number PG103003) is available from Microchip Technology Inc. Factory programming prior to shipment, known as Serialized Quick Turn Programming^{SM} (SQTP^{SM}), is also available. The device is available in die form or packaged in SOIC or PDIP.

Note:	Information provided herein is preliminary and subject to change without notice.

DIE LAYOUT



L1: Antenna Coil A
L2: Antenna Coil B

PAD COORDINATES (MICRONS)

Pad Name	Lower Left X	Lower Left Y	Upper Right X	Upper Right Y	Passivation Openings		Pad Center X	Pad Center Y
					Pad Width	Pad Height		
Ant. Pad A	-610.0	489.2	-521.0	578.2	89	89	-565.5	533.7
Ant. Pad B	-605.0	-579.8	-516.0	-490.8	89	89	-560.5	-535.3
VSS	-605.0	-58.2	-516.0	30.8	89	89	-560.5	-13.7
VDD	463.4	-181.4	552.4	-92.4	89	89	507.9	-136.9
CLK	463.4	496.8	552.4	585.8	89	89	507.9	541.3
VPRG	463.4	157.6	552.4	246.6	89	89	507.9	202.1

Note 1: All coordinates are referenced from the center of the die. The minimum distance between pads (edge to edge) is 10 mil.

2: Die Size = 1.417 mm x 1.513 mm

Serialized Quick Turn Programming (SQTP) is a service mark of Microchip Technology Inc.

5.0 ELECTRICAL CHARACTERISTICS

Storage temperature .....- 65°C to +150°C
Ambient temp. with power applied .....-40°C to +125°C
Maximum current into coil pads .....50 mA

***Notice:** Stresses above those listed under “Maximum ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operational listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

TABLE 5-1: PAD FUNCTION TABLE

Name	Function
Ant. A	Connected to antenna coil L1
Ant. B	Connected to antenna coils L1 and L2
Vss	Connected to antenna coil L2. Device ground during test mode.
VDD	DC voltage supply for programming
CLD	Main clock pulse for device
VPRG	Input/Output for programming and read test.

TABLE 5-2: DC CHARACTERISTICS

All parameters apply across the specified operating ranges, unless otherwise noted.	Commercial (C): Tamb = -20°C to 70°C					
Parameters	**Symbol**	**Min**	**Typ**	**Max**	**Units**	**Conditions**
Reading voltage	VDDR	2.4	—	—	V	VDD voltage for reading
Hysteresis voltage	VHYST	—	TBD	—	TBD	
Operating current	IDDR	—	7	10	µA	VDD = 2.4V during reading at 25°C
Testing voltage	VDDT	—	4	—	V	
Programming voltage: High level input voltage Low level input voltage High voltage	 VIH VIL VHH	 0.7 * VDDT — —	 — — 20	 — 0.3 * VDDT —	 V V V	External DC voltage for programming and testing
Current leakage during sleep time	IDD_OFF	—	10	—	nA	Note
Modulation resistance	RM	—	3	4	Ω	DC resistance between Drain and Source gates of the modulation transistor (when it is turned on)
Pull-Down resistor	RPDW	5	8	—	KΩ	CLK and VPRG internal pull-down resistor

Note: This parameter is not tested in production.

TABLE 5-3: AC CHARACTERISTICS

All parameters apply across the specified operating ranges, unless otherwise noted.	Commercial (C): Tamb = -20°C to 50°C					
Parameters	**Symbol**	**Min**	**Typ**	**Max**	**Units**	**Conditions**
Operating frequency	F_c	13.5598	13.56	13.5602	MHz	Carrier frequency
Modulation frequency	F_M	58	70	82	kHz	Manchester
Coil voltage during reading	VPP_AC	4	—	—	VPP	Peak-to-Peak AC voltage across the coil during reading
Coil clamp voltage	VCLMP_AC	—	32	—	VPP	Peak -to-Peak coil clamp voltage
Test mode clock frequency	F_{clk}		115	500	kHz	25°C
Sleep time	TOFF	60	100	140	ms	Off time for anticollision feature, at 25°C
Internal resonant capacitor (MCRF360)	CRES	90	100	110	pF	Internal resonant capacitor between Antenna Pad A and VSS (at 13.56 MHz)
Resonant frequency (MCRF360)	FR	12.93	13.56	14.30	MHz	with L = 1.377 µH
Write/Erase pulse width	TWC	—	2	10	ms	Time to program bit, at 25°C
Clock high time	THIGH	—	4.4	—	µs	25°C
Clock low time	TLOW	—	4.4	—	µs	25°C
Stop condition pulse width	TPW:STO	—	1000	—	ns	25°C
Stop condition setup time	TSU:STO	—	200	—	ns	25°C
Setup time for high voltage	TSU:HH	—	800	—	ns	25°C
High voltage delay time	TDL:HH	—	800	—	ns	Delay time before the next clock, at 25°C
Data input setup time	TSU:DAT	—	450	—	ns	25°C
Data input hold time	THD:DAT	—	1.2	—	µs	25°C
Output valid from clock	TAA	—	200	—	ns	25°C
Data retention	—	200		—	Years	For T < 120°C

TABLE 5-4: ABSOLUTE MAXIMUM/MINIMUM RATINGS

Parameters	**Symbol**	**Min**	**Max**	**Units**	**Conditions**
Coil current	IPP_AC	—	40	mA	Peak-to-Peak coil current
Maximum Power Dissipation	PMPD	—	1	W	
Assembly temperature	TASM	—	300	°C	< 10 sec
Storage temperature	TSTORE	-65	150	°C	

6.0 FUNCTIONAL DESCRIPTION

The device contains three major sections. The first one is the RF Front-End section, second is the Controller Logic, and third is the Memory section. Figure 6-1 shows the block diagram of the device.

6.1 RF Front-End Section

The RF Front-End section includes power supply, power-on-reset, and data modulation circuits.

6.1.1 POWER SUPPLY

The power supply circuit generates DC voltage (V_{DD}) by rectifying induced AC coil voltage. The power supply circuit includes high-voltage clamping diodes to prevent excessive voltage development across the antenna coil.

6.1.2 POWER-ON-RESET (POR)

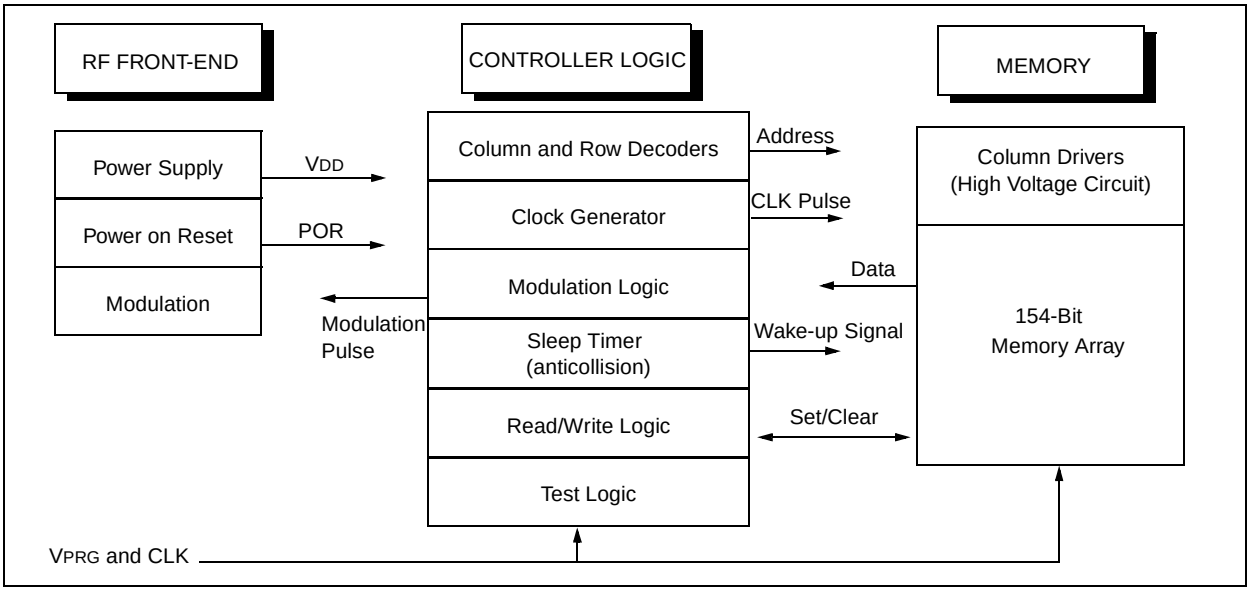
This circuit generates a power-on-reset when the tag first enters the reader field. The reset releases when sufficient power has developed on the V_{DD} regulator to allow for correct operation.

6.1.3 DATA MODULATION

The data modulation circuit consists of a modulation transistor (MOSFET) and a 1-turn antenna coil (L2). The two are connected in parallel. The transistor is designed to result in less than two ohms (R_M) between Antenna Pad B and V_{SS}. As the transistor turns on, the transistor shorts L2 and, therefore, the external LC circuit is detuned (cloaking).

Cloaking and uncloaking occur by driving the transistor on and off, respectively. Therefore, since the data is encoded by a Manchester format, data bit '1' will be sent by uncloaking and cloaking the transistor for 7 µs, each. Similarly, data bit '0' will be sent by cloaking and uncloaking the transistor for 7 µs, each.

FIGURE 6-1: BLOCK DIAGRAM



6.2 Antenna

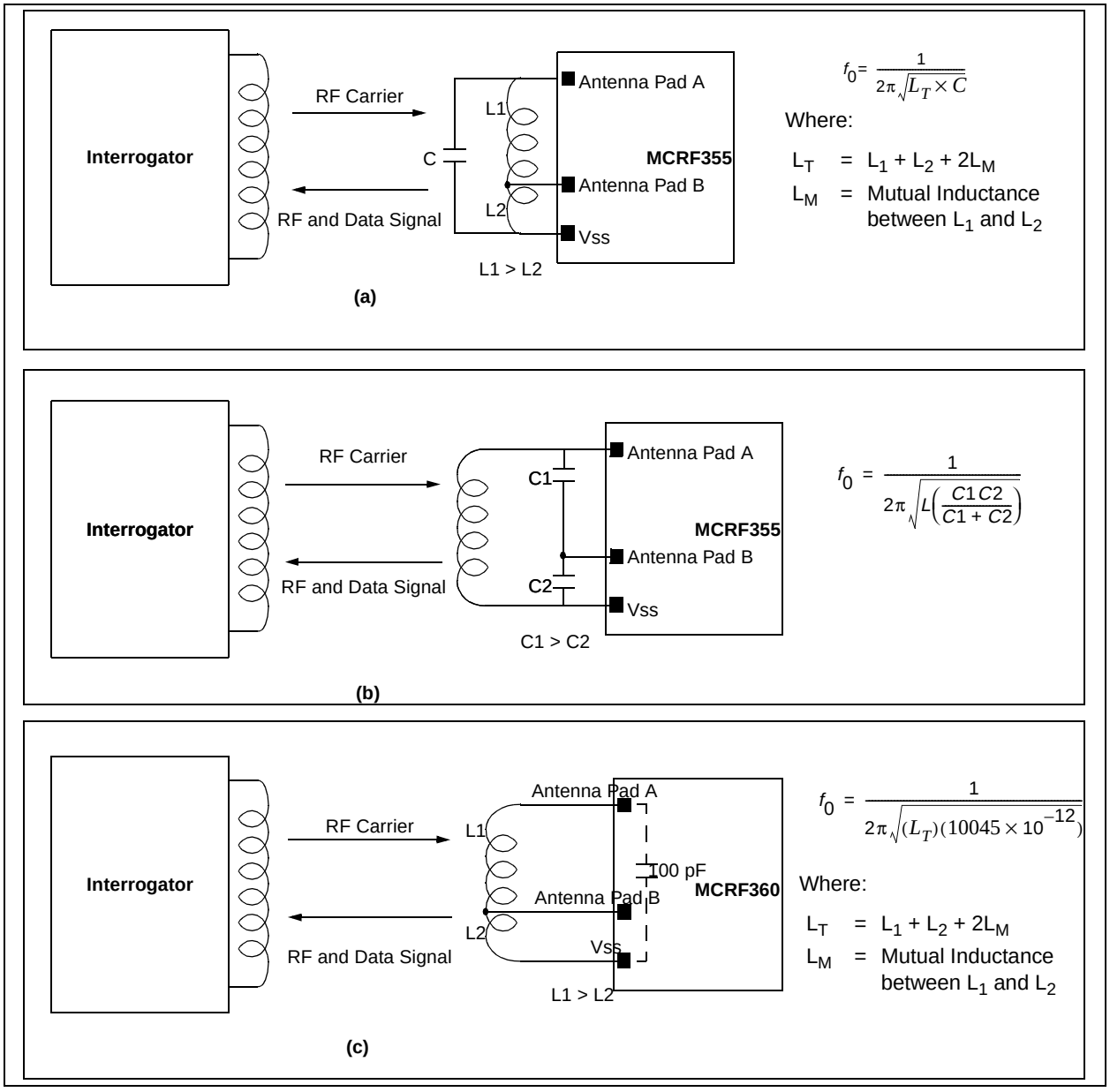
The MCRF360 requires an external inductor and capacitance in order to resonate at 13.56 MHz. About one-fourth of the turns of the inductor should be connected between Antenna Pad B and VSS; remaining turns should be connected between Antenna Pad A and Antenna Pad B. The MCRF355 can use a 1.377 µH inductor plus 100 pF of external capacitance in order to resonate at 13.56 MHz.

Figure 6-2(a) shows a configuration of an external circuit for the MCRF355. Two external antenna coils (L1 and L2) in series and a capacitor that is connected across the two inductors form a parallel resonant circuit to pick up incoming RF signals and also send back modulated signals to the reader. The first coil (L1) is connected between Antenna Pad A and Antenna Pad B. The second coil (L2) is connected between Antenna Pad B and VSS. The capacitor is connected between Antenna Pad A and VSS.

Figure 6-2(b) shows another configuration of an external circuit for the MCRF355. In this case, the resonant circuit is formed by two capacitors (C1 and C2) and one inductor.

Figure 6-2(c) shows a configuration of an external circuit for MCRF360.

FIGURE 6-2: CONFIGURATION OF EXTERNAL RESONANT CIRCUITS



6.3 Controller Logic

6.3.1 CLOCK PULSE GENERATOR

This circuit generates a clock pulse (CLK). The clock pulse is generated by an on-board, time-base oscillator. The clock pulse is used for baud rate timing, data modulation rate, etc.

6.3.2 MODULATION LOGIC

This logic acts upon the serial data (154 bits) being read from the memory array. The data is then converted to Manchester code. The code waveforms are shown in Figure 6-3. The encoded data is then fed to the modulation gate in the RF Front-End section.

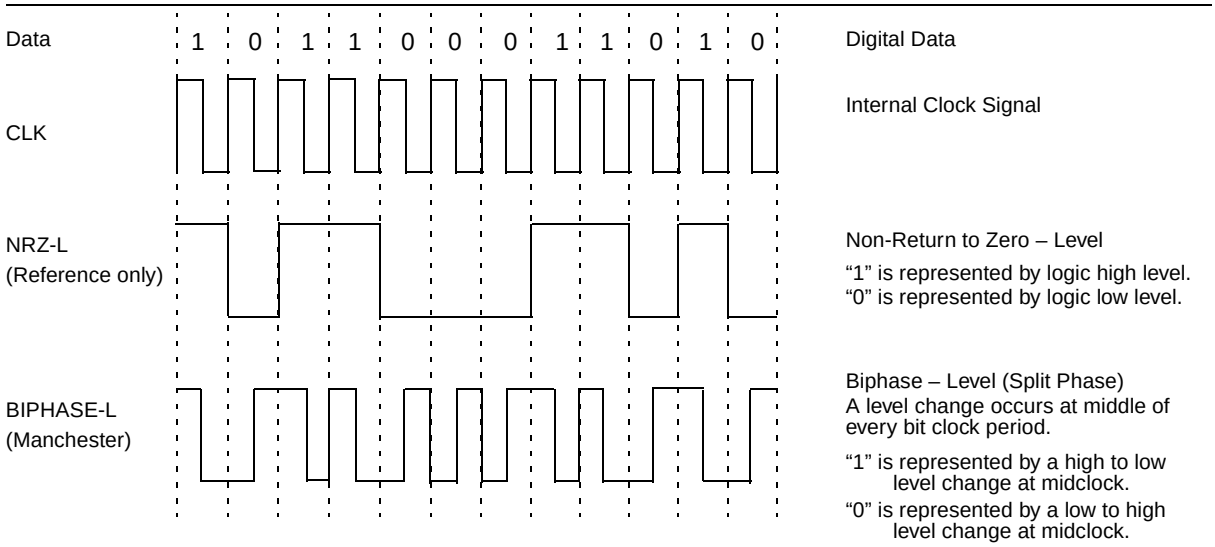
6.3.3 SLEEP TIMER

This circuit generates a sleep time (100 ms ± 40%) for the anticollision feature. During this sleep time (TOFF), the modulation transistor remains in a turned-on condition (cloaked) which detunes the LC resonant circuit away from the operating frequency (13.56 MHz).

6.3.4 READ/WRITE LOGIC

This logic controls the reading and programming of the memory array.

FIGURE 6-3: CODE WAVEFORMS



Note: The CLK and NRZ-L signals are shown for reference only. The NRZ-L is not an output of the device

7.0 DEVICE PROGRAMMING

MCRF355/360 is a contact programmable device. The device has 154 bits of programmable memory. It can be programmed in the following procedure. (A programmer, part number PG103003, is also available from Microchip.)

7.0.1 PROGRAMMING LOGIC

Programming logic is enabled by applying power to the device and clocking the device via the CLK pad while loading the mode code via the VPRG pad (See Examples 7-1 through 7-4 for test definitions). Both the CLK and the VPRG pads have internal pull-down resistors.

7.1 Pin Configuration

Connect antenna pads A, B, and VSS to ground.

7.2 Pin Timing

1. Apply VDDT voltage to VDD. Leave VSS, CLK, and VPRG at ground.
2. Load mode code into the VPRG pad. The VPRG is sampled at CLK low to high edge.
3. The above mode function (3.2.2) will be executed when the last bit of code is entered.
4. Power the device off (VDD = VSS) to exit programming mode.
5. An alternative method to exit the programming mode is to bring CLK logic “High” before VPRG to VHH (high voltage).
6. Any programming mode can be entered after exiting the current function.

7.3 Programming Mode

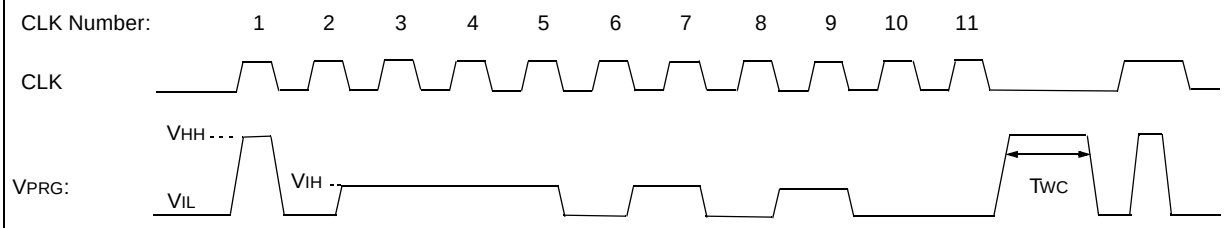
1. Erase EE Code: `0111010100`
2. Program EE Code: `0111010010`
3. Read EE Code: `0111010110`

Note: ‘0’ means logic “Low” (VIL) and ‘1’ means logic “High” (VIH).

7.4 Signal Timing

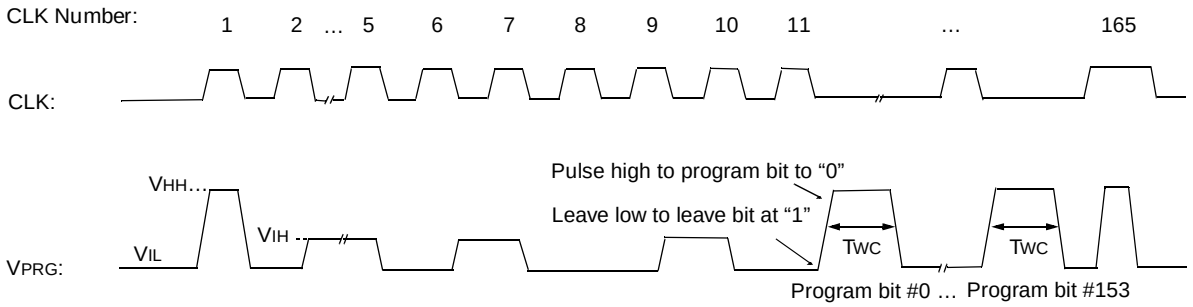
Examples 7-1 through 7-4 show the timing sequence for programming and reading of the device.

EXAMPLE 7-1: PROGRAMMING MODE 1: ERASE EE



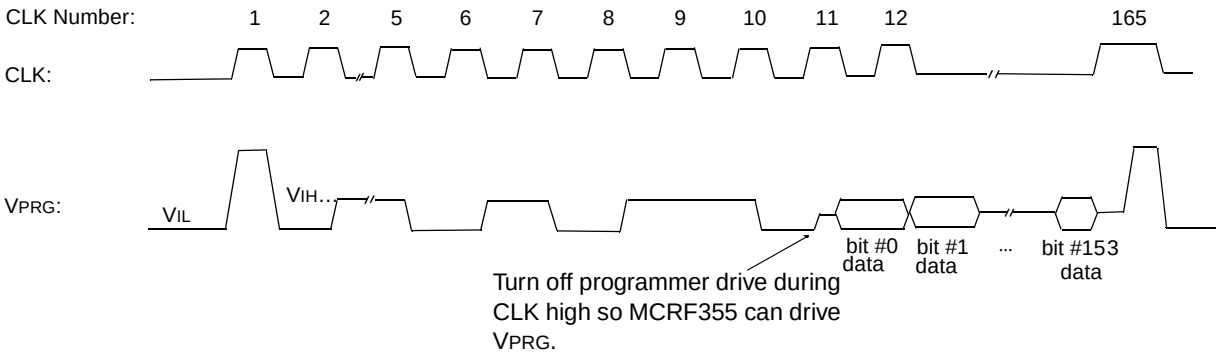
Note: Erases entire array to a ‘1’ state between CLK and Number 11 and 12.

EXAMPLE 7-2: PROGRAMMING MODE 2: PROGRAM EE

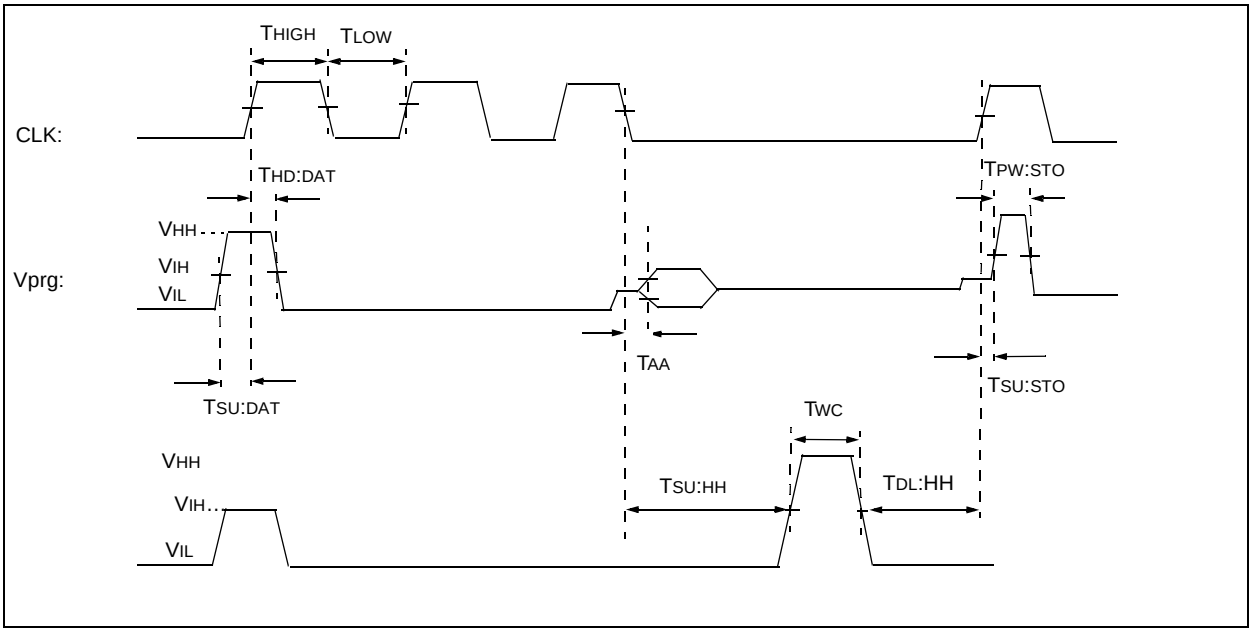


Note: Pulsing VPRG to VHH for the bit programming time while holding the CLK low programs the bit to a ‘0’.

EXAMPLE 7-3: PROGRAMMING MODE 3: READ EE



EXAMPLE 7-4: TIMING DATA



MCRF355/360 GUIDE PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, e.g., on pricing or delivery, please refer to the factory or the listed sales office.

MCRF355 – /WF

Package:		WF	= Sawed wafer on frame (11 mil backgrind)
		WFB	= Bumped, sawed wafer on frame (11 mil backgrind)
		W	= Wafer (11 mil backgrind)
		WB	= Bumped wafer (11 mil backgrind)
		S	= Dice in waffle pack
		SB	= Bumped die in waffle pack
		SN	= 150 mil SOIC
		P	= PDIP
Temperature Range:			= -20°C to +50°C
Part Number:	MCRF355		= 13.56 MHz Anticollision device
	MCRF360		= 13.56 MHz Anticollision device with 100 pF of on-chip resonance capacitance

Sales and Support

Data Sheets

Products supported by a preliminary Data Sheet may have an errata sheet describing minor operational differences and recommended workarounds. To determine if an errata sheet exists for a particular device, please contact one of the following:

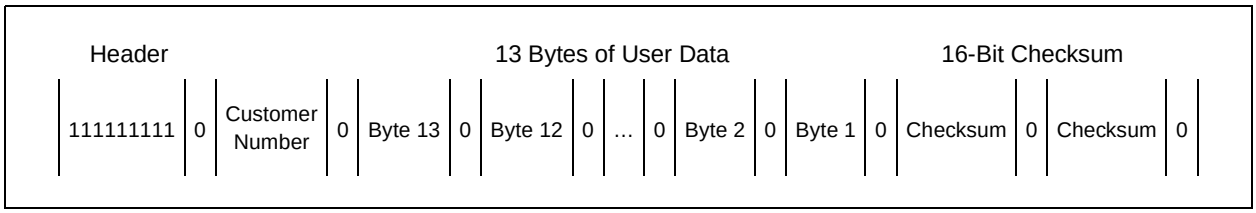
1. Your local Microchip sales office
2. The Microchip Corporate Literature Center U.S. FAX: (480) 786-7277
3. The Microchip Worldwide Site (www.microchip.com)

Please specify which device, revision of silicon and Data Sheet (include Literature #) you are using.

New Customer Notification System

Register on our web site (www.microchip.com/cn) to receive the most current information on our products.

Microchip Development Kit Sample Format



9	bit header
8	bit customer number
104	bits (13 x 8) of user data
17	bits of zeros between each byte, header, and checksum
16	bits of checksum
Total: 154	bits

Notes:

- Users can program all 154 bits of the MCRF355/360. The array can be programmed in any custom format and with any combination of bits.
- The format presented here is used for Microchip microID™ Development System (DV103003) and can be ordered as production material with a unique customer number.
- See TB032 for information on ordering custom programmed production material.
- The Microchip Development System (DV103003) uses nine 1's (111111111) as header.
- The preprogrammed tag samples in the development kit have hex 11(= 0001 0001) as the customer number.
- For the development system, users can program the customer number (1 byte) plus the 13 bytes of user data, or they can deselect the "Microchip Format" option in the MicroID™ RFLAB and program all 154 bits in any format.
- When users program the samples using the MicroID™ RFLAB, the RFLAB calculates the checksum (2 bytes) automatically by adding up all 14 bytes (customer number + 13 bytes of user data), and put into the checksum field in the device memory. See Example 1 for details.
- When the programmed tag is energized by the reader field, the tag outputs all 154 bits of data.
- When the demo reader detects data from the tag, it reports the 14 bytes of the data (customer number plus 13 bytes of user data) to the host computer if the header and checksum are correct. The reader does not send the header and checksum to the host computer.
- The "MicroID™ RFLab" or a simple terminal program such as "terminal.exe" can be used to read the reader's output (28 hex digits) on the host computer.
- When the demo reader is used in the terminal mode ("terminal.exe), the tag's data appear after the first two dummy ASCII characters (GG). See Example 2 for details.

EXAMPLE 7-1: CHECKSUM

Checksum (xxxxxxxx xxxxxxxx) = Byte 1 + Byte 2 + ...+ Byte 13 + Customer Number (1 byte)

EXAMPLE 7-2: READER'S OUTPUT IN TERMINAL MODE ("TERMINAL.EXE")

The demo reader outputs GG+28 hex digits, i.e., GG 12345678901234567890ABCDEFGF.

The first two ASCII characters (GG) are dummy characters.

The tag's data are the next 28 hex digits (112 bits) after the first two ASCII characters (GG).

NOTES:

MCRF355/360 Factory Programming Support (SQTP^{SM})

INTRODUCTION

The MCRF355 and MCRF360 are 13.56 MHz RF tags which can be contact programmed. The contact programming of the device can be performed by the user or factory-programmed by Microchip Technology, Inc. upon customer request. All 154 bits of data may be programmed in any format or pattern defined by the customer.

For factory programming, ID codes and series numbers must be supplied by the customer or an algorithm may be specified by the customer. This technical brief describes only the case in which identification codes (ID) and series numbers are supplied. The customer may supply the ID codes and series numbers on floppy disk or via email. The codes must conform to the Serialized Quick Turn Programming^{SM} (SQTP^{SM}) format below:

FILE SPECIFICATION

SQTP codes supplied to Microchip must comply with the following format:

The ID code file is a plain ASCII text file from floppy disk or email (no headers).

If code files are compressed, they should be self-extracting files.

The code files are used in alphabetical order of their file names (including letters and numbers).

Used (i.e., programmed) code files are discarded by Microchip after use.

Each line of the code file must contain one ID code for one IC.

The code is in hexadecimal format.

The code line is exactly 154 bits (39 hex characters, where the last 2 bits of the last character are don't cares).

Each line must end with a carriage return.

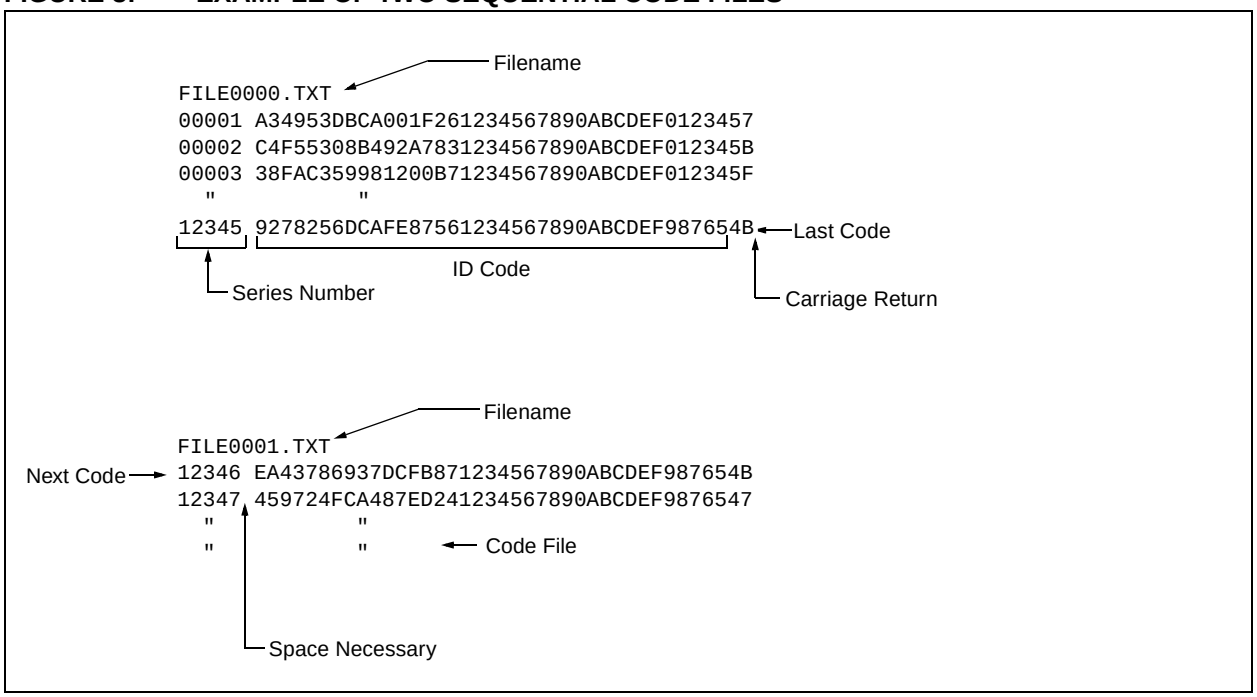
Each hexadecimal ID code must be preceded by a decimal series number.

Series number and ID code must be separated by a space.

The series number must be unique and ascending to avoid double programming.

The series numbers of two consecutive files must also count up for proper linking.

FIGURE 8: EXAMPLE OF TWO SEQUENTIAL CODE FILES



Serialized Quick Turn Programming (SQTP) is a service mark of Microchip Technology inc.

NOTES:

AN707

MCRF 355/360 Applications

Author: *Dr. Youbok Lee, Ph.D.*
Microchip Technology Inc.

INTRODUCTION

The MCRF355 passive RFID device is designed for low cost, multiple reading, and various high volume tagging applications using a frequency band of 13.56 MHz. The device has a total of 154 memory bits that can be reprogrammed by a contact programmer. The device operates with a 70 kHz data rate, and asynchronously with respect to the reader's carrier. The device turns on when the coil voltage reaches 4 VPP and outputs data with a Manchester format (see Figure 2-3 in the data sheet). With the given data rate (70 kHz), it takes about 2.2 ms to transmit all 154 bits of the data. After transmitting all data, the device goes into a sleep mode for 100 ms +/- 50%.

The MCRF355 needs only an external parallel LC resonant circuit that consists of an antenna coil and a capacitor for operation. The external LC components must be connected between antenna A, B, and ground pads. The circuit formed between Antenna Pad A and the ground pad must be tuned to the operating frequency of the reader antenna.

MODE OF OPERATION

The device transmits data by tuning and detuning the resonant frequency of the external circuit. This process is accomplished by using an internal modulation gate (CMOS), that has a very low turn-on resistance (2 ~ 4 ohms) between Drain and Source. This gate turns on during a logic "High" period of the modulation signal and off otherwise. When the gate turns on, its low turn-on resistance shorts the external circuit between Antenna Pad B and the ground pad. Therefore, the resonant frequency of the circuit changes. This is called *detuned* or *cloaking.* Since the detuned tag is out of the frequency band of the reader, the reader can't see it.

The modulation gate turns off as the modulation signal goes to a logic "Low." This turn-off condition again tunes the resonant circuit to the frequency of the reader antenna. Therefore the reader sees the tag again. This is called *tuned* or *uncloaking.*

The tag coil induces maximum voltage during "uncloaking (tuned)" and minimum voltage during cloaking (detuned). Therefore, the cloaking and uncloaking events develop an amplitude modulation signal in the tag coil.

This amplitude modulated signal in the tag coil perturbs the voltage envelope in the reader coil. The reader coil has maximum voltage during cloaking (detuned) and minimum voltage during uncloaking (tuned). By detecting the voltage envelope, the data signal from the tag can be readily reconstructed.

Once the device transmits all 154 bits of data, it goes into "sleep mode" for about 100 ms. The tag wakes up from sleep time (100 ms) and transmits the data package for 2.2 ms and goes into sleep mode again. The device repeats the transmitting and sleep cycles as long as it is energized.

FIGURE 1: VOLTAGE ENVELOPE IN READER COIL

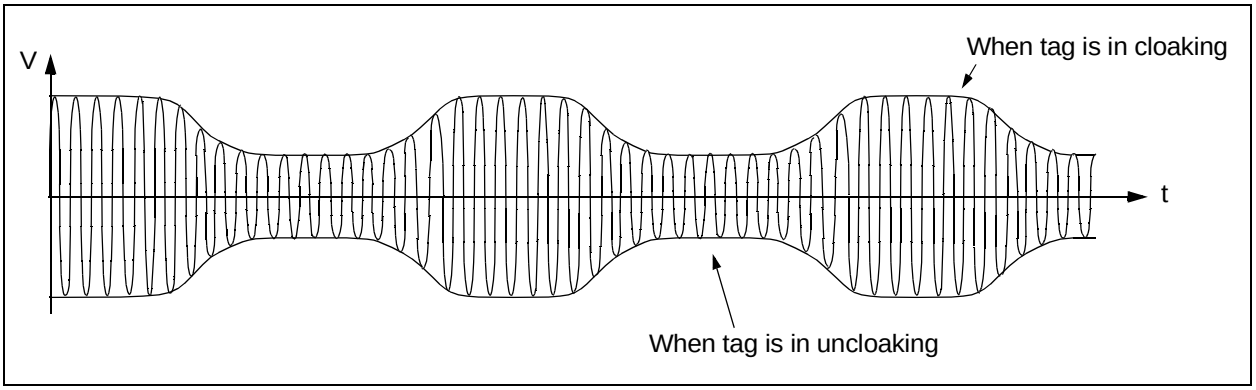
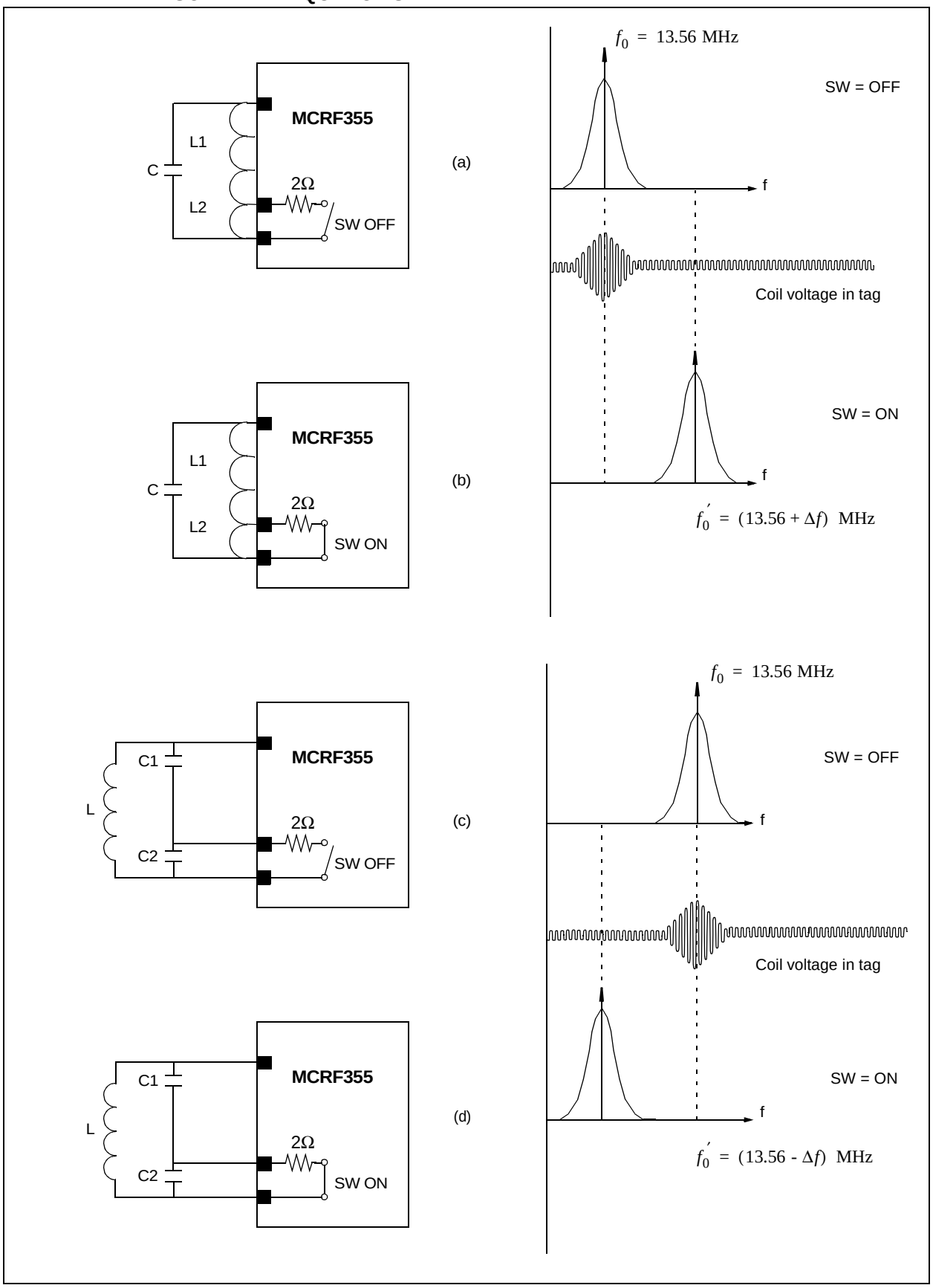


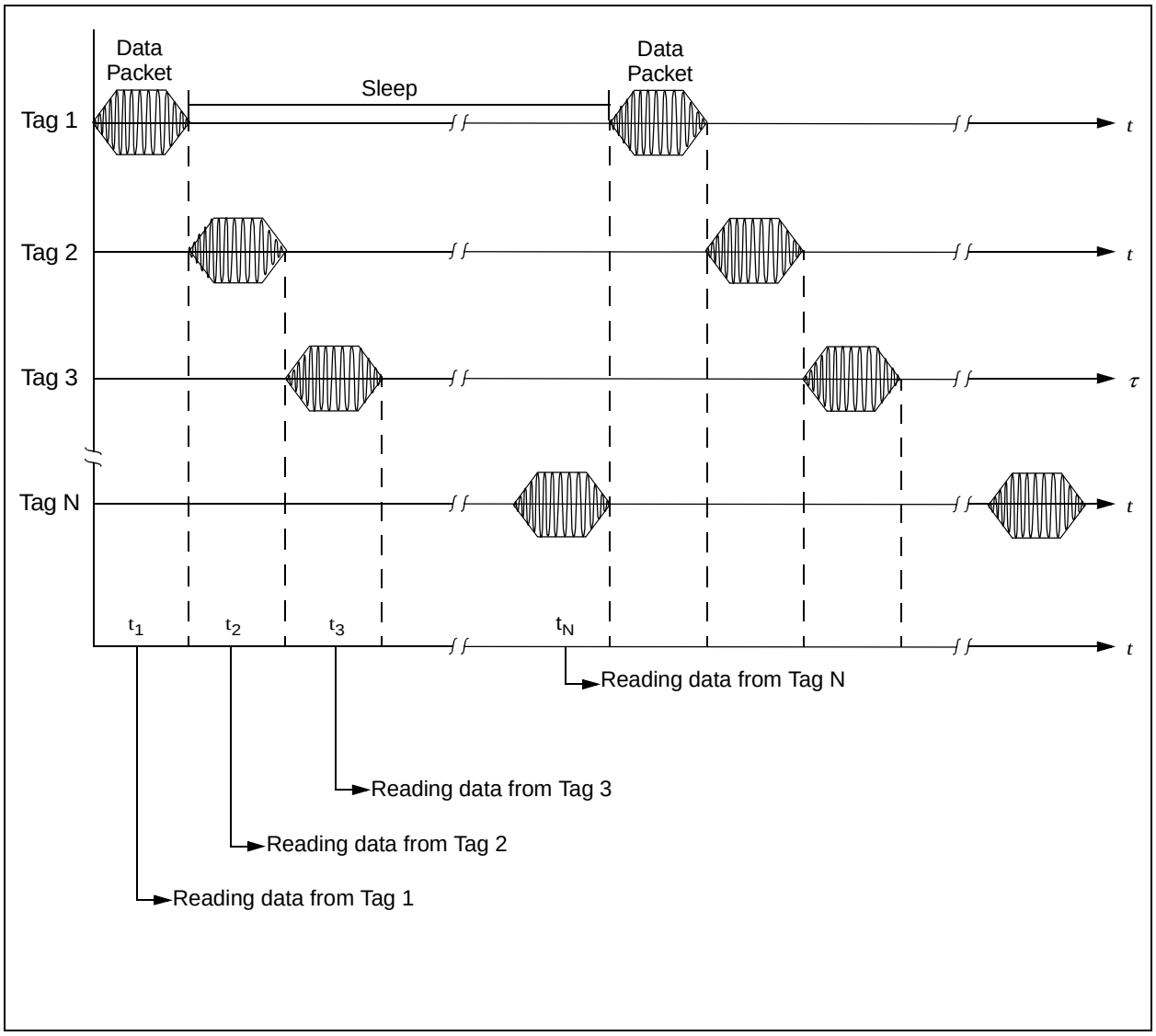
FIGURE 2: (A) UNCLOAKING (TUNED) AND (B) CLOAKING (DETUNED) MODES AND THEIR RESONANT FREQUENCIES



ANTICOLLISION FEATURES

During sleep mode, the device remains in a cloaked state where the circuit is detuned. Therefore, the reader can't see the tag during sleep time. While one tag is in sleep mode, the reader can receive data from other tags. This enables the reader to receive clean data from many tags without any data collision. This ability to read multiple tags in the same RF field is called a*nticollision.* Theoretically, more than 50 tags can be read in the same RF field. However, it is affected by distance from the tag to the reader, angular orientation, movement of the tags, and spacial distribution of the tags.

FIGURE 3: EXAMPLE OF READING MULTIPLE TAGS



EXTERNAL CIRCUIT CONFIGURATION

Since the device transmits data by tuning and detuning the antenna circuit, caution must be given in the external circuit configuration. For a better modulation index, the differences between the tuned and detuned frequencies must be wide enough (about 3 ~ 6 MHz).

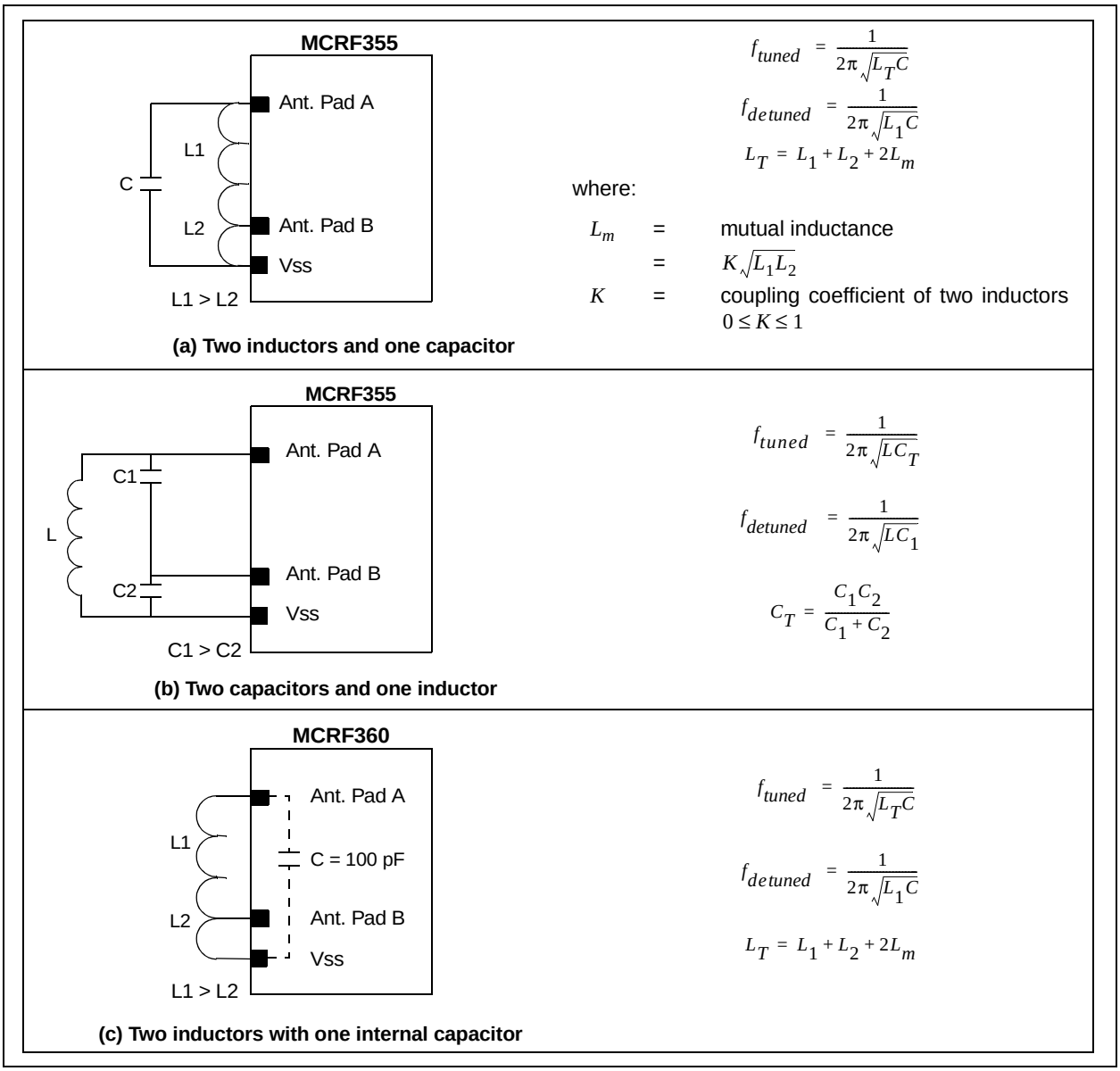
Figure 4 shows various configurations of the external circuit. The choice of the configuration must be chosen depending on the form-factor of the tag. For example, (a) is a better choice for printed circuit tags while, (b) is a better candidate for coil-wound tags. Both (a) and (b) relate to the MCRF355.

In configuration (a), the tuned resonance frequency is determined by a total capacitance and inductance from Antenna Pad A to Vss. During cloaking, the internal switch (modulation gate) shorts Antenna Pad B and Vss. Therefore, the inductance L2 is shorted out. As a result, the detuned frequency is determined by the total capacitance and inductance L1. When shorting the inductance between Antenna Pad B and Vss, the detuned (cloak) frequency is higher than the tuned (uncloak) frequency

In configuration (b), the tuned frequency (uncloak) is determined by the inductance L and the total capacitance between Antenna Pad A and Vss. The circuit detunes (cloak) when C_2 is shorted. This detuned frequency (cloak) is lower than the tuned (uncloak) frequency

The MCRF360 includes a 100 pF internal capacitor. This device needs only an external inductor for operation. The explanation on tuning and detuning is the same as for configuration (a).

FIGURE 4: VARIOUS EXTERNAL CIRCUIT CONFIGURATIONS



$$f_{tuned} = \frac{1}{2\pi\sqrt{L_T C}}$$

$$f_{detuned} = \frac{1}{2\pi\sqrt{L_1 C}}$$

$$L_T = L_1 + L_2 + 2L_m$$

where:

L_m = mutual inductance

= $K\sqrt{L_1 L_2}$

K = coupling coefficient of two inductors $0 \le K \le 1$

$$f_{tuned} = \frac{1}{2\pi\sqrt{LC_T}}$$

$$f_{detuned} = \frac{1}{2\pi\sqrt{LC_1}}$$

$$C_T = \frac{C_1 C_2}{C_1 + C_2}$$

$$f_{tuned} = \frac{1}{2\pi\sqrt{L_T C}}$$

$$f_{detuned} = \frac{1}{2\pi\sqrt{L_1 C}}$$

$$L_T = L_1 + L_2 + 2L_m$$

PROGRAMMING OF DEVICE

All of the memory bits in the MCRF355/360 are reprogrammable by a contact programmer or by factory programming prior to shipment, known as Serialized Quick Turn Programming^{SM} (SQTP^{SM}). For more information about contact programming, see page 69 of the *microID™ 13.56 MHz System Design Guide* (DS21299). For information about SQTP programming, please see TB032 (DS91032), page 19 of the design guide.

Serial Quick Turn Programming (SQTP) is a Service Mark of Microchip Technology Inc.

NOTES:

AN710

Antenna Circuit Design

Author: *Dr. Youbok Lee, Ph.D.*
Microchip Technology Inc.

INTRODUCTION

Passive RFID tags utilize an induced antenna coil voltage for operation. This induced AC voltage is rectified to provide a voltage source for the device. As the DC voltage reaches a certain level, the device starts operating. By providing an energizing RF signal, a reader can communicate with a remotely located device that has no external power source such as a battery. Since the energizing and communication between the reader and tag is accomplished through antenna coils, it is important that the device must be equipped with a proper antenna circuit for successful RFID applications.

An RF signal can be radiated effectively if the linear dimension of the antenna is comparable with the wavelength of the operating frequency. However, the wavelength at 13.56 MHz is 22.12 meters. Therefore, it is difficult to form a true antenna for most RFID applications. Alternatively, a small loop antenna circuit that is resonating at the frequency is used. A current flowing into the coil radiates a near-field magnetic field that falls off with r^{-3}. This type of antenna is called a *magnetic dipole antenna.*

For 13.56 MHz passive tag applications, a few microhenries of inductance and a few hundred pF of resonant capacitor are typically used. The voltage transfer between the reader and tag coils is accomplished through inductive coupling between the two coils. As in a typical transformer, where a voltage in the primary coil transfers to the secondary coil, the voltage in the reader antenna coil is transferred to the tag antenna coil and vice versa. The efficiency of the voltage transfer can be increased significantly with high *Q* circuits.

This section is written for RF coil designers and RFID system engineers. It reviews basic electromagnetic theories on antenna coils, a procedure for coil design, calculation and measurement of inductance, an antenna tuning method, and read range in RFID applications.

REVIEW OF A BASIC THEORY FOR RFID ANTENNA DESIGN

Current and Magnetic Fields

Ampere's law states that current flowing in a conductor produces a magnetic field around the conductor. The magnetic field produced by a current element, as shown in Figure 1, on a round conductor (wire) with a finite length is given by:

EQUATION 1:

$$B_\phi = \frac{\mu_o I}{4\pi r}(\cos\alpha_2 - \cos\alpha_1) \qquad (\text{Weber}/m^2)$$

where:

I = current
r = distance from the center of wire
μ_0 = permeability of free space and given as $4\pi \times 10^{-7}$ (Henry/meter)

In a special case with an infinitely long wire where:

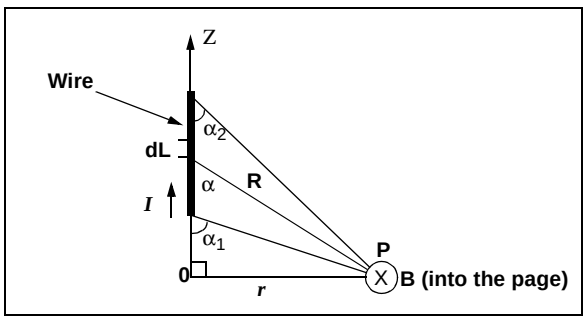
α_1 = -180°
α_2 = 0°

Equation 1 can be rewritten as:

EQUATION 2:

$$B_\phi = \frac{\mu_o I}{2\pi r} \qquad (\text{Weber}/m^2)$$

FIGURE 1: CALCULATION OF MAGNETIC FIELD B AT LOCATION P DUE TO CURRENT I ON A STRAIGHT CONDUCTING WIRE



The magnetic field produced by a circular loop antenna is given by:

EQUATION 3:

$$B_z = \frac{\mu_o I N a^2}{2(a^2 + r^2)^{3/2}}$$

$$= \frac{\mu_o I N a^2}{2}\left(\frac{1}{r^3}\right) \quad \text{for } r^2 >> a^2$$

where

I	=	current
a	=	radius of loop
r	=	distance from the center of wire
μ_0	=	permeability of free space and given as $\mu_o = 4\pi \times 10^{-7}$ (Henry/meter)

The above equation indicates that the magnetic field strength decays with $1/r^3$. A graphical demonstration is shown in Figure 3. It has maximum amplitude in the plane of the loop and directly proportional to both the current and the number of turns, N.

Equation 3 is often used to calculate the ampere-turn requirement for read range. A few examples that calculate the ampere-turns and the field intensity necessary to power the tag will be given in the following sections.

FIGURE 2: CALCULATION OF MAGNETIC FIELD B AT LOCATION P DUE TO CURRENT I ON THE LOOP

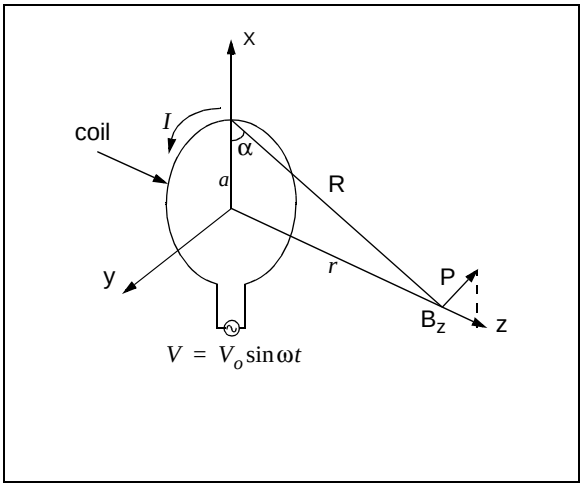
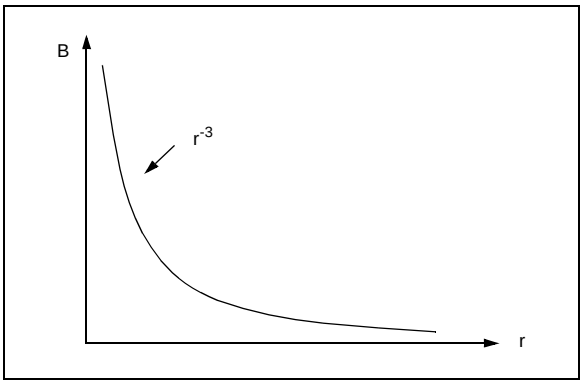


FIGURE 3: DECAYING OF THE MAGNETIC FIELD B VS. DISTANCE *r*



INDUCED VOLTAGE IN AN ANTENNA COIL

Faraday's law states that a time-varying magnetic field through a surface bounded by a closed path induces a voltage around the loop.

Figure 4 shows a simple geometry of an RFID application. When the tag and reader antennas are in close proximity, the time-varying magnetic field B that is produced by a reader antenna coil induces a voltage (called electromotive force or simply EMF) in the closed tag antenna coil. The induced voltage in the coil causes a flow of current on the coil. This is called Faraday's law. The induced voltage on the tag antenna coil is equal to the time rate of change of the magnetic flux Ψ.

EQUATION 4:

$$V = -N\frac{d\Psi}{dt}$$

where:

N	=	number of turns in the antenna coil
Ψ	=	magnetic flux through each turn

The negative sign shows that the induced voltage acts in such a way as to oppose the magnetic flux producing it. This is known as Lenz's Law and it emphasizes the fact that the direction of current flow in the circuit is such that the induced magnetic field produced by the induced current will oppose the original magnetic field.

The magnetic flux Ψ in Equation 4 is the total magnetic field B that is passing through the entire surface of the antenna coil, and found by:

EQUATION 5:

$$\psi = \int B \cdot dS$$

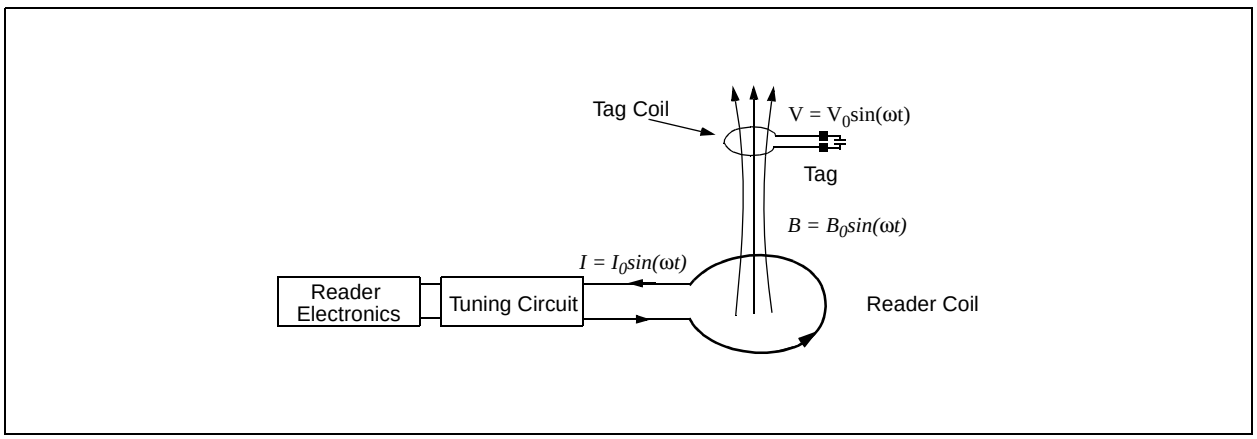
where:

B	=	magnetic field given in Equation 2
S	=	surface area of the coil
•	=	inner product (*cosine angle between two vectors*) of vectors B and surface area S

Note:	Both magnetic field B and surface S are vector quantities.

The presentation of inner product of two vectors in Equation 5 suggests that the total magnetic flux ψ that is passing through the antenna coil is affected by an orientation of the antenna coils. The inner product of two vectors becomes maximized when the cosine angle between the two are 90 degree, or the two (B field and the surface of coil) are perpendicular to each other. The maximum magnetic flux that is passing through the tag coil is obtained when the two coils (reader coil and tag coil) are placed in parallel with respect to each other. This condition results in maximum induced voltage in the tag coil and also maximum read range. The inner product expression in Equation 5 also can be expressed in terms of a mutual coupling between the reader and tag coils. The mutual coupling between the two coils is maximized in the above condition.

FIGURE 4: A BASIC CONFIGURATION OF READER AND TAG ANTENNAS IN RFID APPLICATIONS



Using Equations 3 and 5, Equation 4 can be rewritten as:

EQUATION 6:

$$V = - N_2 \frac{d\Psi_{21}}{dt} = - N_2 \frac{d}{dt}\left(\int B \cdot dS\right)$$

$$= - N_2 \frac{d}{dt}\left[\int \frac{\mu_o i_1 N_1 a^2}{2(a^2 + r^2)^{3/2}} \cdot dS\right]$$

$$= - \left[\frac{\mu_o N_1 N_2 a^2 (\pi b^2)}{2(a^2 + r^2)^{3/2}}\right] \frac{di_1}{dt}$$

$$= - M \frac{di_1}{dt}$$

where:

V	=	voltage in the tag coil
i_1	=	current on the reader coil
a	=	radius of the reader coil
b	=	radius of tag coil
r	=	distance between the two coils
M	=	mutual inductance between the tag and reader coils, and given by:

EQUATION 7:

$$M = \left[\frac{\mu_o \pi N_1 N_2 (ab)^2}{2(a^2 + r^2)^{3/2}}\right]$$

The above equation is equivalent to a voltage transformation in typical transformer applications. The current flow in the primary coil produces a magnetic flux that causes a voltage induction at the secondary coil.

As shown in Equation 6, the tag coil voltage is largely dependent on the mutual inductance between the two coils. The mutual inductance is a function of coil geometry and the spacing between them. The induced voltage in the tag coil decreases with r^{-3}. Therefore, the read range also decreases in the same way.

From Equations 4 and 5, a generalized expression for induced voltage V_o in a tuned loop coil is given by:

EQUATION 8:

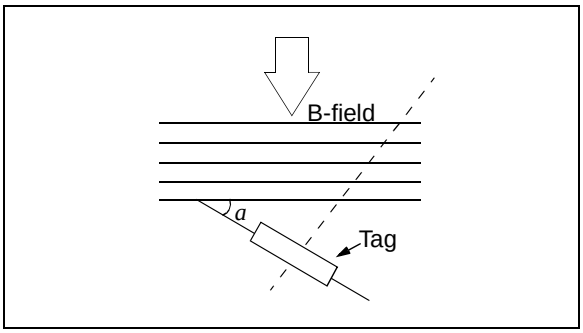
$$V_0 = 2\pi f N S Q B_o \cos\alpha$$

where:

f	=	frequency of the arrival signal
N	=	number of turns of coil in the loop
S	=	area of the loop in square meters (m^2)
Q	=	quality factor of circuit
B_0	=	strength of the arrival signal
α	=	angle of arrival of the signal

In the above equation, the quality factor Q is a measure of the selectivity of the frequency of the interest. The Q will be defined in Equations 31 through 47.

FIGURE 5: ORIENTATION DEPENDENCY OF THE TAG ANTENNA



The induced voltage developed across the loop antenna coil is a function of the angle of the arrival signal. The induced voltage is maximized when the antenna coil is placed in parallel with the incoming signal where $\alpha = 0$.

EXAMPLE 1: CALCULATION OF B-FIELD IN A TAG COIL

The MCRF355 device turns on when the antenna coil develops 4 VPP across it. This voltage is rectified and the device starts to operate when it reaches 2.4 VDC. The B-field to induce a 4 VPP coil voltage with an ISO standard 7810 card size (85.6 x 54 x 0.76 mm) is calculated from the coil voltage equation using Equation 8.

EQUATION 9:

$$V_o = 2\pi f N S Q B_o \cos\alpha = 4$$

and

$$B_o = \frac{4/(\sqrt{2})}{2\pi f N S Q \cos\alpha} = 0.0449 \qquad (\mu wbm^{-2})$$

where the following parameters are used in the above calculation:

Tag coil size	=	(85.6 x 54) mm^2 (ISO card size) = 0.0046224 m^2
Frequency	=	13.56 MHz
Number of turns	=	4
Q of tag antenna coil	=	40
AC coil voltage to turn on the tag	=	4 VPP
$\cos\alpha$ =	=	1 (normal direction, $\alpha = 0$).

EXAMPLE 2: NUMBER OF TURNS AND CURRENT (AMPERE-TURNS)

Assuming that the reader should provide a read range of 15 inches (38.1 cm) for the tag given in the previous example, the current and number of turns of a reader antenna coil is calculated from Equation 3:

EQUATION 10:

$$(NI)_{rms} = \frac{2B_z(a^2 + r^2)^{3/2}}{\mu a^2}$$

$$= \frac{2(0.0449 \times 10^{-6})(0.1^2 + (0.38)^2)^{3/2}}{(4\pi \times 10^{-7})(0.1^2)}$$

$$= 0.43(\text{ampere - turns})$$

The above result indicates that it needs a 430 mA for 1 turn coil, and 215 mA for 2-turn coil.

EXAMPLE 3: OPTIMUM COIL DIAMETER OF THE READER COIL

An optimum coil diameter that requires the minimum number of ampere-turns for a particular read range can be found from Equation 3 such as:

EQUATION 11:

$$NI = K\frac{(a^2 + r^2)^{\frac{3}{2}}}{a^2}$$

where: $K = \frac{2B_z}{\mu_o}$

By taking derivative with respect to the radius a,

$$\frac{d(NI)}{da} = K\ \frac{3/2(a^2 + r^2)^{1/2}(2a^3) - 2a(a^2 + r^2)^{3/2}}{a^4}$$

$$= K\frac{(a^2 - 2r^2)(a^2 + r^2)^{1/2}}{a^3}$$

The above equation becomes minimized when:

$$a^2 - 2r^2 = 0$$

The above result shows a relationship between the read range vs. optimum coil diameter. The optimum coil diameter is found as:

EQUATION 12:

$$a = \sqrt{2}r$$

where:

a = radius of coil

r = read range.

The result indicates that the optimum loop radius, a, is 1.414 times the demanded read range r.

WIRE TYPES AND OHMIC LOSSES

Wire Size and DC Resistance

The diameter of electrical wire is expressed as the American Wire Gauge (AWG) number. The gauge number is inversely proportional to diameter, and the diameter is roughly doubled every six wire gauges. The wire with a smaller diameter has a higher DC resistance. The DC resistance for a conductor with a uniform cross-sectional area is found by:

EQUATION 13:

$$R_{DC} = \frac{l}{\sigma S} \qquad (\Omega)$$

where:

l = total length of the wire
σ = conductivity
S = cross-sectional area

Table 1 shows the diameter for bare and enamel-coated wires, and DC resistance.

AC Resistance of Wire

At DC, charge carriers are evenly distributed through the entire cross section of a wire. As the frequency increases, the reactance near the center of the wire increases. This results in higher impedance to the current density in the region. Therefore, the charge moves away from the center of the wire and towards the edge of the wire. As a result, the current density decreases in the center of the wire and increases near the edge of the wire. This is called a *skin effect*. The depth into the conductor at which the current density falls to 1/e, or 37% of its value along the surface, is known as the *skin depth* and is a function of the frequency and the permeability and conductivity of the medium. The skin depth is given by:

EQUATION 14:

$$\delta = \frac{1}{\sqrt{\pi f \mu \sigma}}$$

where:

f = frequency
μ = permeability of material
σ = conductivity of the material

EXAMPLE 4:

The skin depth for a copper wire at 13.56 MHz can be calculated as:

EQUATION 15:

$$\delta = \frac{1}{\sqrt{\pi f (4\pi \times 10^{-7})(5.8 \times 10^{-7})}}$$

$$= \frac{0.0179}{\sqrt{f}} \qquad (m)$$

$$= 0.187 \qquad (mm)$$

The wire resistance increases with frequency, and the resistance due to the skin depth is called an AC resistance. An approximated formula for the AC resistance is given by:

EQUATION 16:

$$R_{ac} \approx \frac{1}{2\sigma\pi\delta} = (R_{DC})\frac{a}{2\delta} \qquad (\Omega)$$

where:

a = coil radius

TABLE 1: AWG WIRE CHART

Wire Size (AWG)	Dia. in Mils (bare)	Dia. in Mils (coated)	Ohms/ 1000 ft.	Cross Section (mils)
1	289.3	—	0.126	83690
2	287.6	—	0.156	66360
3	229.4	—	0.197	52620
4	204.3	—	0.249	41740
5	181.9	—	0.313	33090
6	162.0	—	0.395	26240
7	166.3	—	0.498	20820
8	128.5	131.6	0.628	16510
9	114.4	116.3	0.793	13090
10	101.9	106.2	0.999	10380
11	90.7	93.5	1.26	8230
12	80.8	83.3	1.59	6530
13	72.0	74.1	2.00	5180
14	64.1	66.7	2.52	4110
15	57.1	59.5	3.18	3260
16	50.8	52.9	4.02	2580
17	45.3	47.2	5.05	2060
18	40.3	42.4	6.39	1620
19	35.9	37.9	8.05	1290
20	32.0	34.0	10.1	1020
21	28.5	30.2	12.8	812
22	25.3	28.0	16.2	640
23	22.6	24.2	20.3	511
24	20.1	21.6	25.7	404
25	17.9	19.3	32.4	320
26	15.9	17.2	41.0	253
27	14.2	15.4	51.4	202
28	12.6	13.8	65.3	159
29	11.3	12.3	81.2	123
30	10.0	11.0	106.0	100
31	8.9	9.9	131	79.2
32	8.0	8.8	162	64.0
33	7.1	7.9	206	50.4
34	6.3	7.0	261	39.7
35	5.6	6.3	331	31.4
36	5.0	5.7	415	25.0
37	4.5	5.1	512	20.2
38	4.0	4.5	648	16.0
39	3.5	4.0	847	12.2
40	3.1	3.5	1080	9.61
41	2.8	3.1	1320	7.84
42	2.5	2.8	1660	6.25
43	2.2	2.5	2140	4.84
44	2.0	2.3	2590	4.00
45	1.76	1.9	3350	3.10
46	1.57	1.7	4210	2.46
47	1.40	1.6	5290	1.96
48	1.24	1.4	6750	1.54
49	1.11	1.3	8420	1.23
50	0.99	1.1	10600	0.98

Note: 1 mil = 2.54 x 10^{-3} cm

Note: 1 mil = 2.54 x 10^{-3} cm

INDUCTANCE OF VARIOUS ANTENNA COILS

An electric current element that flows through a conductor produces a magnetic field. This time-varying magnetic field is capable of producing a flow of current through another conductor – this is called *inductance*. The inductance L depends on the physical characteristics of the conductor. A coil has more inductance than a straight wire of the same material, and a coil with more turns has more inductance than a coil with fewer turns. The inductance L of inductor is defined as the ratio of the total magnetic flux linkage to the current I through the inductor:

EQUATION 17:

$$L = \frac{N\Psi}{I} \qquad \text{(Henry)}$$

where:

N = number of turns
I = current
Ψ = the magnetic flux

For a coil with multiple turns, the inductance is greater as the spacing between turns becomes smaller. Therefore, the tag antenna coil that has to be formed in a limited space often needs a multilayer winding to reduce the number of turns.

Calculation of Inductance

Inductance of the coil can be calculated in many different ways. Some are readily available from references[1-4]. It must be remembered that for RF coils the actual resulting inductance may differ from the calculated true result because of distributed capacitance. For that reason, inductance calculations are generally used only for a starting point in the final design.

Inductance of a Straight Wound Wire

The inductance of a straight wound wire shown in Figure 1 is given by:

EQUATION 18:

$$L = 0.002l\left[\log_e\frac{2l}{a} - \frac{3}{4}\right] \qquad (\mu H)$$

where:

l and a = length and radius of wire in cm, respectively.

EXAMPLE 5: INDUCTANCE CALCULATION FOR A STRAIGHT WIRE:

The inductance of a wire with 10 feet (304.8cm) long and 2 mm in diameter is calculated as follows:

EQUATION 19:

$$L = 0.002(304.8)\left[\ln\left(\frac{2(304.8)}{0.1}\right) - \frac{3}{4}\right]$$

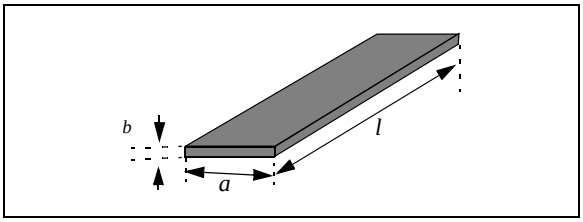
$$= 0.60967(7.965)$$

$$= 4.855(\mu H)$$

Inductance of Thin Film Inductor with a Rectangular Cross Section

Inductance of a conductor with rectangular cross section as shown in Figure 6 is calculated as:

FIGURE 6: A STRAIGHT THIN FILM INDUCTOR



EQUATION 20:

$$L = 0.002l\left\{\ln\left(\frac{2l}{a+b}\right) + 0.50049 + \frac{a+b}{3l}\right\} \qquad (\mu H)$$

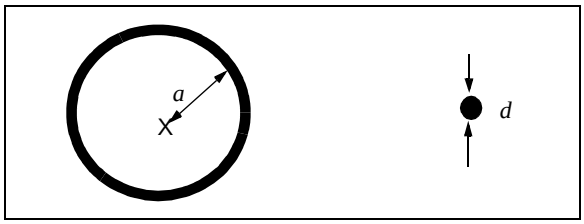
where:

a = width in cm
b = thickness in cm
l = length of conductor in cm

Inductance of a Circular Coil with Single Turn

The inductance of a circular coil shown in Figure 7 can be calculated by:

FIGURE 7: A CIRCULAR COIL WITH SINGLE TURN



EQUATION 21:

$$L = 0.01257(a)\left[2.303\log_{10}\left(\frac{16a}{d} - 2\right)\right] \qquad (\mu H)$$

where:

- a = mean radius of loop in (cm)
- d = diameter of wire in (cm)

Inductance of an N-turn Circular Coil with Single Layer

The inductance of a circular coil with single layer is calculated as:

EQUATION 22:

$$L = \frac{(aN)^2}{22.9l + 25.4a} \qquad (\mu H)$$

where:

- N = number of turns
- l = length
- a = the radius of coil in cm

Inductance of N-turn Circular Coil with Multilayer

FIGURE 8: N-TURN CIRCULAR COIL WITH SINGLE LAYER

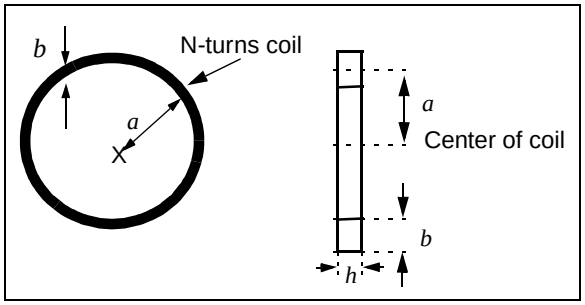


Figure 8 shows an N-turn inductor of circular coil with multilayer. Its inductance is calculated by:

EQUATION 23:

$$L = \frac{0.31(aN)^2}{6a + 9h + 10b} \qquad (\mu H)$$

where:

- a = average radius of the coil in cm
- N = number of turns
- b = winding thickness in cm
- h = winding height in cm

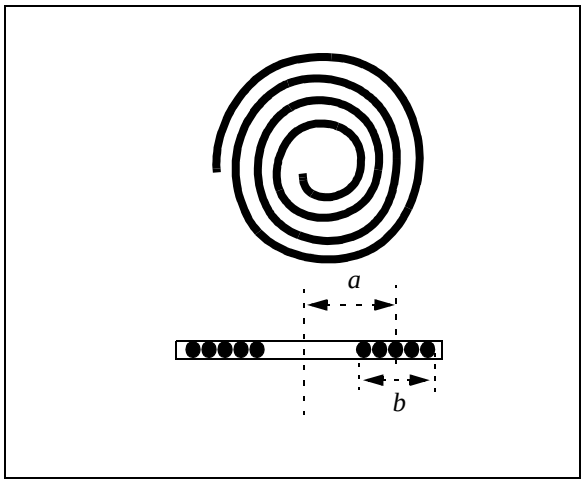
Inductance of Spiral Wound Coil with Single Layer

The inductance of a spiral inductor is calculated by:

EQUATION 24:

$$L = \frac{(aN)^2}{8a + 11b} \qquad (\mu H)$$

FIGURE 9: A SPIRAL COIL



Inductance of N-turn Square Loop Coil with Multilayer

Inductance of a multilayer square loop coil is calculated by:

EQUATION 25:

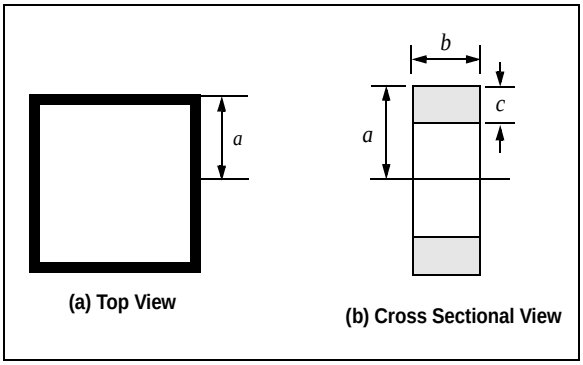
$$L = 0.008aN^2\left\{2.303\log_{10}\left(\frac{a}{b+c}\right) + 0.2235\frac{b+c}{a} + 0.726\right\}(\mu H)$$

where:

- N = number of turns
- a = side of square measured to the center of the rectangular cross section of winding
- b = winding length
- c = winding depth as shown in Figure 10.

Note: All dimensions are in cm.

FIGURE 10: N-TURN SQUARE LOOP COIL WITH MULTILAYER



Inductance of a Flat Square Coil

Inductance of a flat square coil of rectangular cross section with N turns is calculated by[4]:

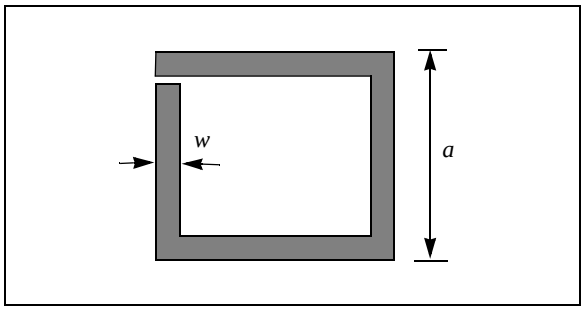
EQUATION 26:

$$L = 0.0467aN^2\left\{\log_{10}\left(2\frac{a^2}{t+w}\right) - \log_{10}(2.414a)\right\} + 0.02032aN^2\left\{0.914 + \left[\frac{0.2235}{a}(t+w)\right]\right\}$$

where:

L	=	in µH
a	=	side length in inches
t	=	thickness in inches
w	=	width in inches

FIGURE 11: SQUARE LOOP INDUCTOR WITH A RECTANGULAR CROSS SECTION



The formulas for inductance are widely published and provide a reasonable approximation for the relationship between inductance and the number of turns for a given physical size[1–4]. When building prototype coils, it is wise to exceed the number of calculated turns by about 10% and then remove turns to achieve a right value. For production coils, it is best to specify an inductance and tolerance rather than a specific number of turns.

CONFIGURATION OF ANTENNA CIRCUITS

Reader Antenna Circuits

The inductance for the reader antenna coil for 13.56 MHz is typically in the range of a few microhenries (µH). The antenna can be formed by aircore or ferrite core inductors. The antenna can also be formed by a metallic or conductive trace on PCB board or on flexible substrate.

The reader antenna can be made of either a single coil, that is typically forming a series or a parallel resonant circuit, or a double loop (transformer) antenna coil. Figure 12 shows various configurations of reader antenna circuit. The coil circuit must be tuned to the operating frequency to maximize power efficiency. The tuned LC resonant circuit is the same as the bandpass filter that passes only a selected frequency. The Q of the tuned circuit is related to both read range and bandwidth of the circuit. More on this subject will be discussed in the following section.

Choosing the size and type of antenna circuit depends on the system design topology. The series resonant circuit results in minimum impedance at the resonance frequency. Therefore, it draws a maximum current at the resonance frequency. Because of its simple circuit topology and relatively low cost, this type of antenna circuit is suitable for proximity reader antenna.

On the other hand, a parallel resonant circuit results in maximum impedance at the resonance frequency. Therefore, maximum voltage is available at the resonance frequency. Although it has a minimum resonant current, it still has a strong circulating current that is proportional to Q of the circuit. The double loop antenna coil that is formed by two parallel antenna circuits can also be used.

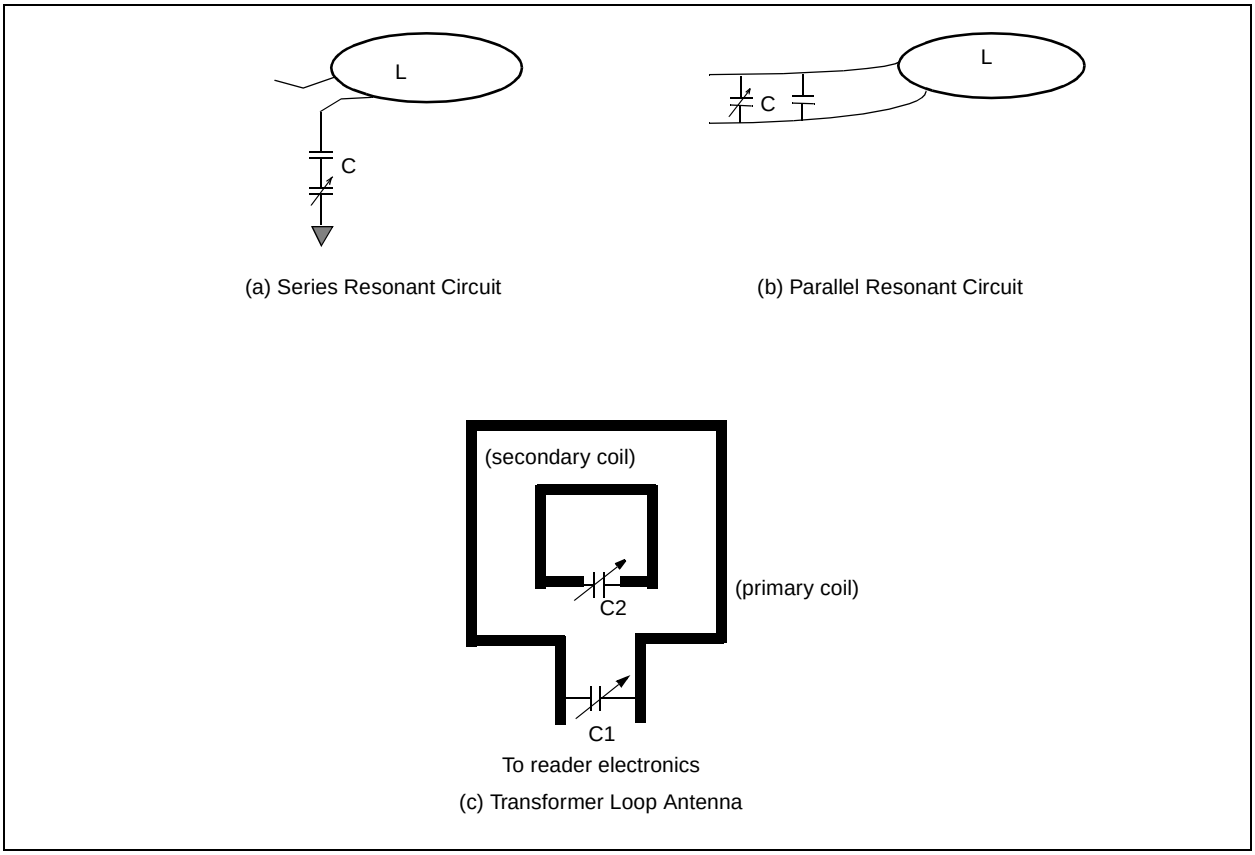
The frequency tolerance of the carrier frequency and output power level from the read antenna is regulated by government regulations (e.g., FCC in the USA).

FCC limits for 13.56 MHz frequency band are as follows:

1. Tolerance of the carrier frequency: 13.56 MHz +/- 0.01% = +/- 1.356 kHz.
2. Frequency bandwidth: +/- 7 kHz.
3. Power level of fundamental frequency: 10 mv/m at 30 meters from the transmitter.
4. Power level for harmonics: -50.45 dB down from the fundamental signal.

The transmission circuit including the antenna coil must be designed to meet the FCC limits.

FIGURE 12: VARIOUS READER ANTENNA CIRCUITS



Tag Antenna Circuits

The MCRF355 device communicates data by tuning and detuning the antenna circuit (see AN707). Figure 13 shows examples of the external circuit arrangement.

The external circuit must be tuned to the resonant frequency of the reader antenna. In a detuned condition, a circuit element between the antenna B and VSS pads is shorted. The frequency difference (delta frequency) between tuned and detuned frequencies must be adjusted properly for optimum operation. It has been found that maximum modulation index and maximum read range occur when the tuned and detuned frequencies are separated by 3 to 6 MHz.

The tuned frequency is formed from the circuit elements between the antenna A and VSS pads without shorting the antenna B pad. The detuned frequency is found when the antenna B pad is shorted. This detuned frequency is calculated from the circuit between antenna A and VSS pads excluding the circuit element between antenna B and VSS pads.

In Figure 13 (a), the tuned resonant frequency is

EQUATION 27:

$$f_o = \frac{1}{2\pi\sqrt{L_T C}}$$

where:

- L_T = $L_1 + L_2 + 2L_M$ = Total inductance between antenna A and VSS pads
- L_1 = inductance between antenna A and antenna B pads
- L_2 = inductance between ant. B and VSS pads
- M = mutual inductance between coil 1 and coil 2
  = $k\sqrt{L_1 L_2}$
- k = coupling coefficient between the two coils
- C = tuning capacitance

and detuned frequency is

EQUATION 28:

$$f_{detuned} = \frac{1}{2\pi\sqrt{L_1 C}}$$

In this case, $f_{detuned}$ is higher than f_{tuned}.

Figure 13(b) shows another example of the external circuit arrangement. This configuration controls C_2 for tuned and detuned frequencies. The tuned and untuned frequencies are

EQUATION 29:

$$f_{tuned} = \frac{1}{2\pi\sqrt{\left(\frac{C_1 C_2}{C_1 + C_2}\right)L}}$$

and

EQUATION 30:

$$f_{detuned} = \frac{1}{2\pi\sqrt{LC_1}}$$

A typical inductance of the coil is about a few microhenry with a few turns. Once the inductance is determined, the resonant capacitance is calculated from the above equations. For example, if a coil has an inductance of 1.3 µH, then it needs a 106 pF of capacitance to resonate at 13.56 MHz.

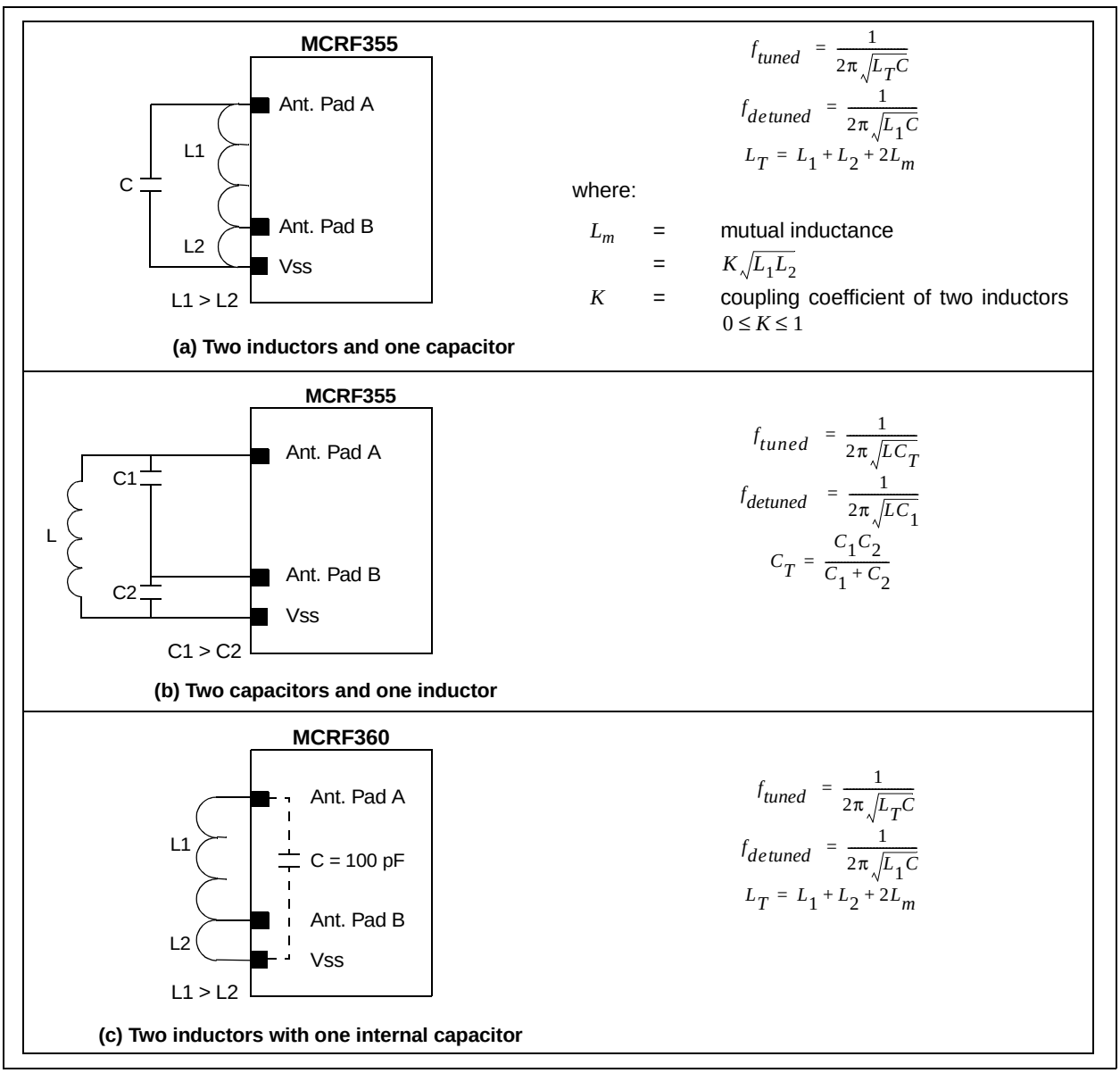
CONSIDERATION ON QUALITY FACTOR *Q* AND BANDWIDTH OF TUNING CIRCUIT

The voltage across the coil is a product of quality factor *Q* of the circuit and input voltage. Therefore, for a given input voltage signal, the coil voltage is directly proportional to the *Q* of the circuit. In general, a higher *Q* results in longer read range. However, the *Q* is also related to the bandwidth of the circuit as shown in the following equation.

EQUATION 31:

$$Q = \frac{f_o}{B}$$

FIGURE 13: VARIOUS EXTERNAL CIRCUIT CONFIGURATIONS



(a) Two inductors and one capacitor

(b) Two capacitors and one inductor

(c) Two inductors with one internal capacitor

Bandwidth requirement and limit on circuit *Q* for MCRF355

Since the MCRF355 operates with a data rate of 70 kHz, the reader antenna circuit needs a bandwidth of at least twice of the data rate. Therefore, it needs:

EQUATION 32:

$$B_{minimum} = 140 \text{ kHz}$$

Assuming the circuit is turned at 13.56 MHz, the maximum attainable Q is obtained from Equations 31 and 32:

EQUATION 33:

$$Q_{max} = \frac{f_o}{B} = 96.8$$

In a practical LC resonant circuit, the range of Q for 13.56 MHz band is about 40. However, the Q can be significantly increased with a ferrite core inductor. The system designer must consider the above limits for optimum operation.

RESONANT CIRCUITS

Once the frequency and the inductance of the coil are determined, the resonant capacitance can be calculated from:

EQUATION 34:

$$C = \frac{1}{L(2\pi f_o)^2}$$

In practical applications, parasitic (distributed) capacitance is present between turns. The parasitic capacitance in a typical tag antenna coil is a few (pF). This parasitic capacitance increases with operating frequency of the device.

There are two different resonant circuits: parallel and series. The parallel resonant circuit has maximum impedance at the resonance frequency. It has a minimum current and maximum voltage at the resonance frequency. Although the current in the circuit is minimum at the resonant frequency, there are a circulation current that is proportional to Q of the circuit. The parallel resonant circuit is used in both the tag and the high-power reader antenna circuit.

On the other hand, the series resonant circuit has a minimum impedance at the resonance frequency. As a result, maximum current is available in the circuit. Because of its simplicity and the availability of the high current into the antenna element, the series resonant circuit is often used for a simple proximity reader.

Parallel Resonant Circuit

Figure 14 shows a simple parallel resonant circuit. The total impedance of the circuit is given by:

EQUATION 35:

$$Z(j\omega) = \frac{j\omega L}{(1-\omega^2 LC) + j\frac{\omega L}{R}} \quad (\Omega)$$

where ω is an angular frequency given as $\omega = 2\pi f$.

The maximum impedance occurs when the denominator in the above equation is minimized. This condition occurs when:

EQUATION 36:

$$\omega^2 LC = 1$$

This is called a resonance condition, and the resonance frequency is given by:

EQUATION 37:

$$f_0 = \frac{1}{2\pi\sqrt{LC}}$$

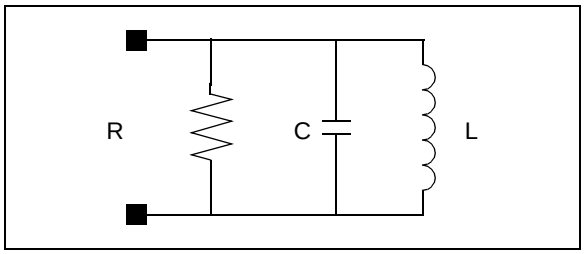
By applying Equation 36 into Equation 35, the impedance at the resonance frequency becomes:

EQUATION 38:

$$Z = R$$

where R is the load resistance.

FIGURE 14: PARALLEL RESONANT CIRCUIT



The R and C in the parallel resonant circuit determine the bandwidth, B, of the circuit.

EQUATION 39:

$$B = \frac{1}{2\pi RC} \qquad (Hz)$$

The quality factor, Q, is defined by various ways such as

EQUATION 40:

$$Q = \frac{\text{Energy Stored in the System per One Cycle}}{\text{Energy Dissipated in the System per One Cycle}}$$

$$= \frac{reactance}{resistance}$$

$$= \frac{\omega L}{r} \qquad \text{For inductance}$$

$$= \frac{1}{\omega cr} \qquad \text{For capacitance}$$

$$= \frac{f_0}{B}$$

where:

ω = $2\pi f$ = angular frequency
f_o = resonant frequency
B = bandwidth
r = ohmic losses

By applying Equation 37 and Equation 39 into Equation 40, the Q in the parallel resonant circuit is:

EQUATION 41:

$$Q = R\sqrt{\frac{C}{L}}$$

The Q in a parallel resonant circuit is proportional to the load resistance R and also to the ratio of capacitance and inductance in the circuit.

When this parallel resonant circuit is used for the tag antenna circuit, the voltage drop across the circuit can be obtained by combining Equations 8 and 41:

EQUATION 42: .

$$V_o = 2\pi f_o NQSB_o \cos\alpha$$

$$= 2\pi f_0 N\left(R\sqrt{\frac{C}{L}}\right)SB_0 \cos\alpha$$

The above equation indicates that the induced voltage in the tag coil is inversely proportional to the square root of the coil inductance, but proportional to the number of turns and surface area of the coil.

Series Resonant Circuit

A simple series resonant circuit is shown in Figure 15. The expression for the impedance of the circuit is:

EQUATION 43:

$$Z(j\omega) = r + j(X_L - X_C) \qquad (\Omega)$$

where:

r = a dc ohmic resistance of coil and capacitor
X_L and X_C = the reactance of the coil and capacitor, respectively, such that:

EQUATION 44:

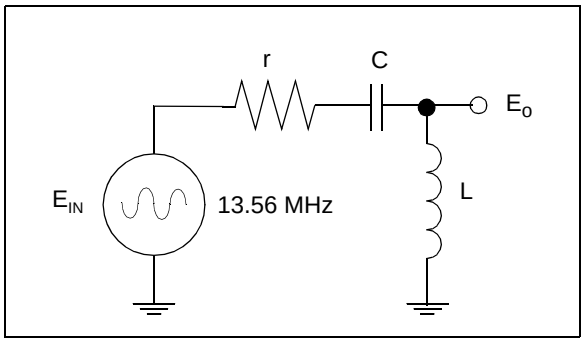
$$X_L = 2\pi f_o L \qquad (\Omega)$$

EQUATION 45:

$$X_c = \frac{1}{2\pi f_o C} \qquad (\Omega)$$

The impedance in Equation 43 becomes minimized when the reactance component cancelled out each other such that $X_L = X_C$. This is called a resonance condition. The resonance frequency is same as the parallel resonant frequency given in Equation 37.

FIGURE 15: SERIES RESONANCE CIRCUIT



The half power frequency bandwidth is determined by r and L, and given by:

EQUATION 46:

$$B = \frac{r}{2\pi L} \qquad (Hz)$$

The quality factor, Q, in the series resonant circuit is given by:

$$Q = \frac{f_0}{B} = \frac{\omega L}{r} = \frac{1}{r\omega C}$$

The series circuit forms a voltage divider, the voltage drops in the coil is given by:

EQUATION 47:

$$V_o = \frac{jX_L}{r + jX_L - jX_c} V_{in}$$

When the circuit is tuned to a resonant frequency such as $X_L = X_C$, the voltage across the coil becomes:

EQUATION 48:

$$V_o = \frac{jX_L}{r} V_{in}$$

$$= jQV_{in}$$

The above equation indicates that the coil voltage is a product of input voltage and Q of the circuit. For example, a circuit with Q of 40 can have a coil voltage that is 40 times higher than input signal. This is because all energy in the input signal spectrum becomes squeezed into a single frequency band.

EXAMPLE 6: CIRCUIT PARAMETERS

If the DC ohmic resistance r is 5 Ω, then the L and C values for 13.56 MHz resonant circuit with Q = 40 are:

EQUATION 49:

$$X_L = Qr_s = 200\Omega$$

$$L = \frac{X_L}{2\pi f} = \frac{200}{2\pi(13.56MHz)} = 2.347 \qquad (\mu H)$$

$$C = \frac{1}{2\pi f X_L} = \frac{1}{2\pi(13.56\ \text{MHz})(200)} = 58.7\ \text{(pF)}$$

TUNING METHOD

The circuit must be tuned to the resonance frequency for a maximum performance (read range) of the device. Two examples of tuning the circuit are as follows:

- **Voltage Measurement Method:**
  a) Set up a voltage signal source at the resonance frequency.
  b) Connect a voltage signal source across the resonant circuit.
  c) Connect an Oscilloscope across the resonant circuit.
  d) Tune the capacitor or the coil while observing the signal amplitude on the Oscilloscope.
  e) Stop the tuning at the maximum voltage.

- **S-parameter or Impedance Measurement Method using Network Analyzer:**
  a) Set up an S-Parameter Test Set (Network Analyzer) for S11 measurement, and do a calibration.
  b) Measure the S11 for the resonant circuit.
  c) Reflection impedance or reflection admittance can be measured instead of the S11.
  d) Tune the capacitor or the coil until a maximum null (S11) occurs at the resonance frequency, f_o. For the impedance measurement, the maximum peak will occur for the parallel resonant circuit, and minimum peak for the series resonant circuit.

FIGURE 16: VOLTAGE VS. FREQUENCY FOR RESONANT CIRCUIT

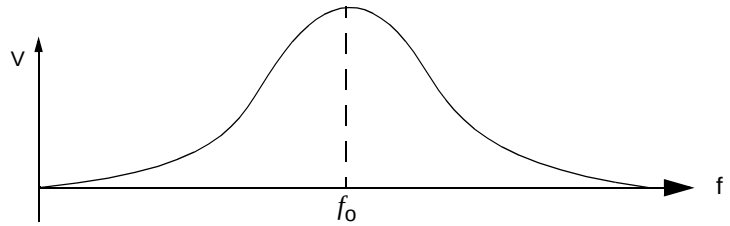
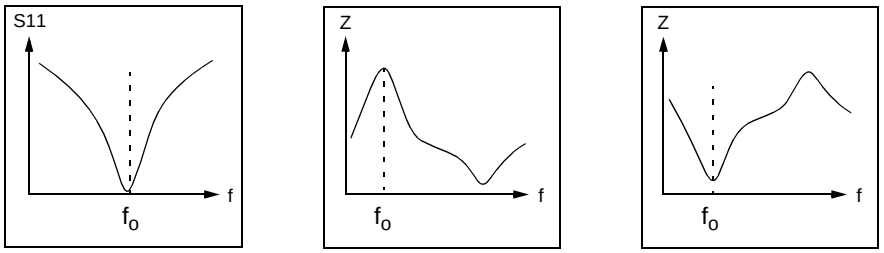


FIGURE 17: FREQUENCY RESPONSES FOR RESONANT CIRCUIT



(a) (b) (c)

Note 1: (a) S11 Response, (b) Impedance Response for a Parallel Resonant Circuit, and (c) Impedance Response for a Series Resonant Circuit.

2: In (a), the null at the resonance frequency represents a minimum input reflection at the resonance frequency. This means the circuit absorbs the signal at the frequency while other frequencies are reflected back. In (b), the impedance curve has a peak at the resonance frequency. This is because the parallel resonant circuit has a maximum impedance at the resonance frequency. (c) shows a response for the series resonant circuit. Since the series resonant circuit has a minimum impedance at the resonance frequency, a minimum peak occurs at the resonance frequency.

READ RANGE OF RFID DEVICES

Read range is defined as a maximum communication distance between the reader and tag. In general, the read range of passive RFID products varies, depending on system configuration and is affected by the following parameters:

a) Operating frequency and performance of antenna coils
b) *Q* of antenna and tuning circuit
c) Antenna orientation
d) Excitation current
e) Sensitivity of receiver
f) Coding (or modulation) and decoding (or demodulation) algorithm
g) Number of data bits and detection (interpretation) algorithm
h) Condition of operating environment (electrical noise), etc.

The read range of 13.56 MHz is relatively longer than that of 125 kHz device. This is because the antenna efficiency increases as the frequency increases. With a given operating frequency, the conditions (a – c) are related to the antenna configuration and tuning circuit. The conditions (d – e) are determined by a circuit topology of reader. The condition (f) is a communication protocol of the device, and (g) is related to a firmware software program for data detection.

Assuming the device is operating under a given condition, the read range of the device is largely affected by the performance of the antenna coil. It is always true that a longer read range is expected with the larger size of the antenna with a proper antenna design. Figures 18 and 19 show typical examples of the read range of various passive RFID devices.

FIGURE 18: READ RANGE VS. TAG SIZE FOR TYPICAL PROXIMITY APPLICATIONS*

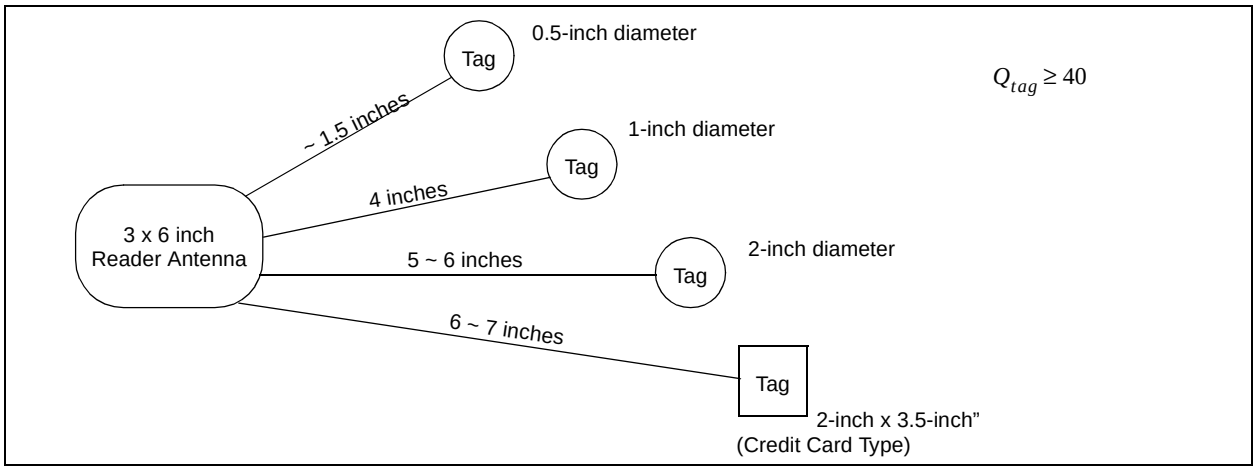
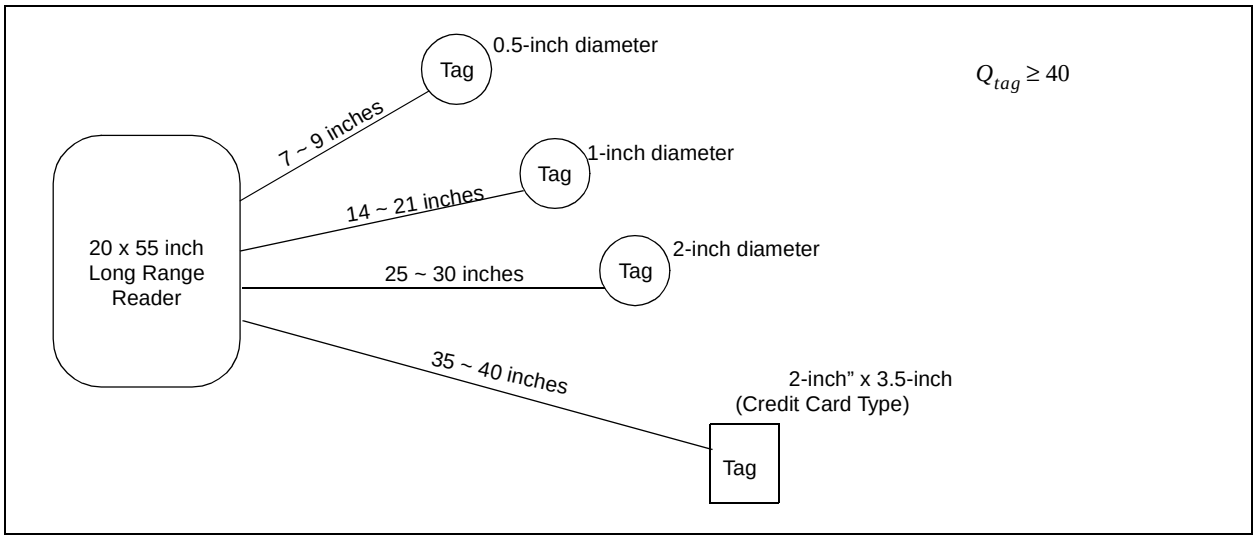


FIGURE 19: READ RANGE VS. TAG SIZE FOR TYPICAL LONG RANGE APPLICATIONS*



Note: Actual results may be shorter or longer than the range shown, depending upon factors discussed above.

REFERENCES

[1] V. G. Welsby, The Theory and Design of Inductance Coils, John Wiley and Sons, Inc., 1960.

[2] Frederick W. Grover, Inductance Calculations Working Formulas and Tables, Dover Publications, Inc., New York, NY., 1946.

[3] Keith Henry, Editor, Radio Engineering Handbook, McGraw-Hill Book Company, New York, NY., 1963.

[4] James K. Hardy, High Frequency Circuit Design, Reston Publishing Company, Inc.Reston, Virginia, 1975.

13.56 MHz Reader Reference Design

1.0 INTRODUCTION

This chapter provides a reference guide for the 13.56 MHz reader designer. The schematic included in this chapter is for the 13.56 MHz Reference Reader included in the DV103003 microID™ Developer's Kit. The circuit is designed for short read-range applications. The basic design can be modified for long-range or other applications with MCRF355/360 devices. An electronic copy of the PICmicro® microcontroller source code is available upon request.

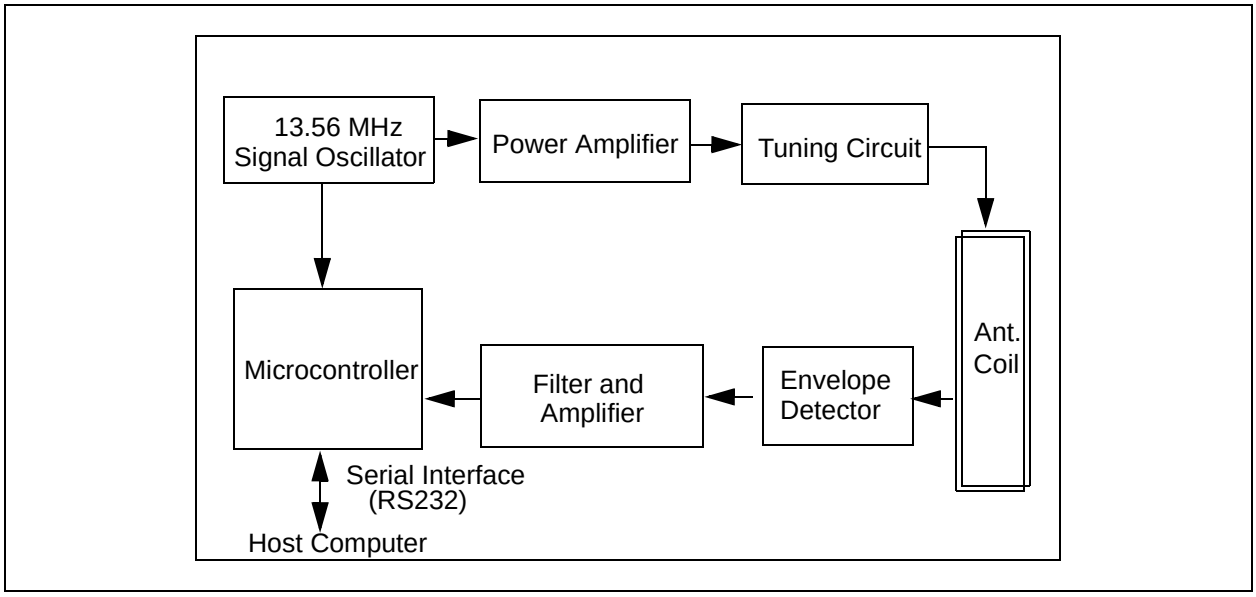
2.0 READER CIRCUITS

The RFID reader consists of transmitting and receiving sections. It transmits a carrier signal (13.56 MHz), receives the backscattered signal from the tag, and performs data processing. The reader also communicates with an external host computer. A basic block diagram of a typical RFID reader is shown in Figure 2-1.

The transmitting section contains a 13.56 MHz signal oscillator (74HC04), power amplifier (Q2), and RF tuning circuits. The tuning circuit matches impedance between the antenna coil circuit and the power driver at 13.56 MHz. The radiating signal strength from the antenna must comply with government regulations. For best performance, the antenna coil circuit must be tuned to the same frequency of the tag. The design for antenna circuits is given in Application Note AN710 (DS00710).

The receiving section contains an envelope detector (D6), hi-pass filters, and amplifiers (U2 and U3). When the tag is energized, it transmits 154 bits of data that is encoded in Biphase-L (Manchester). In the Manchester encoding, data '1' is represented by a logic high-to-low level change at midclock, and data '0' is represented by a low-to-high level change at midclock. There is always a level change at middle of every bit clock.

FIGURE 2-1: FUNCTIONAL BLOCK DIAGRAM OF TYPICAL RFID READER



microID is a trademark of Microchip Technology Inc.
PICmicro is a registered trademark of Microchip Technology In.

FIGURE 2-2: SIGNAL WAVEFORMS

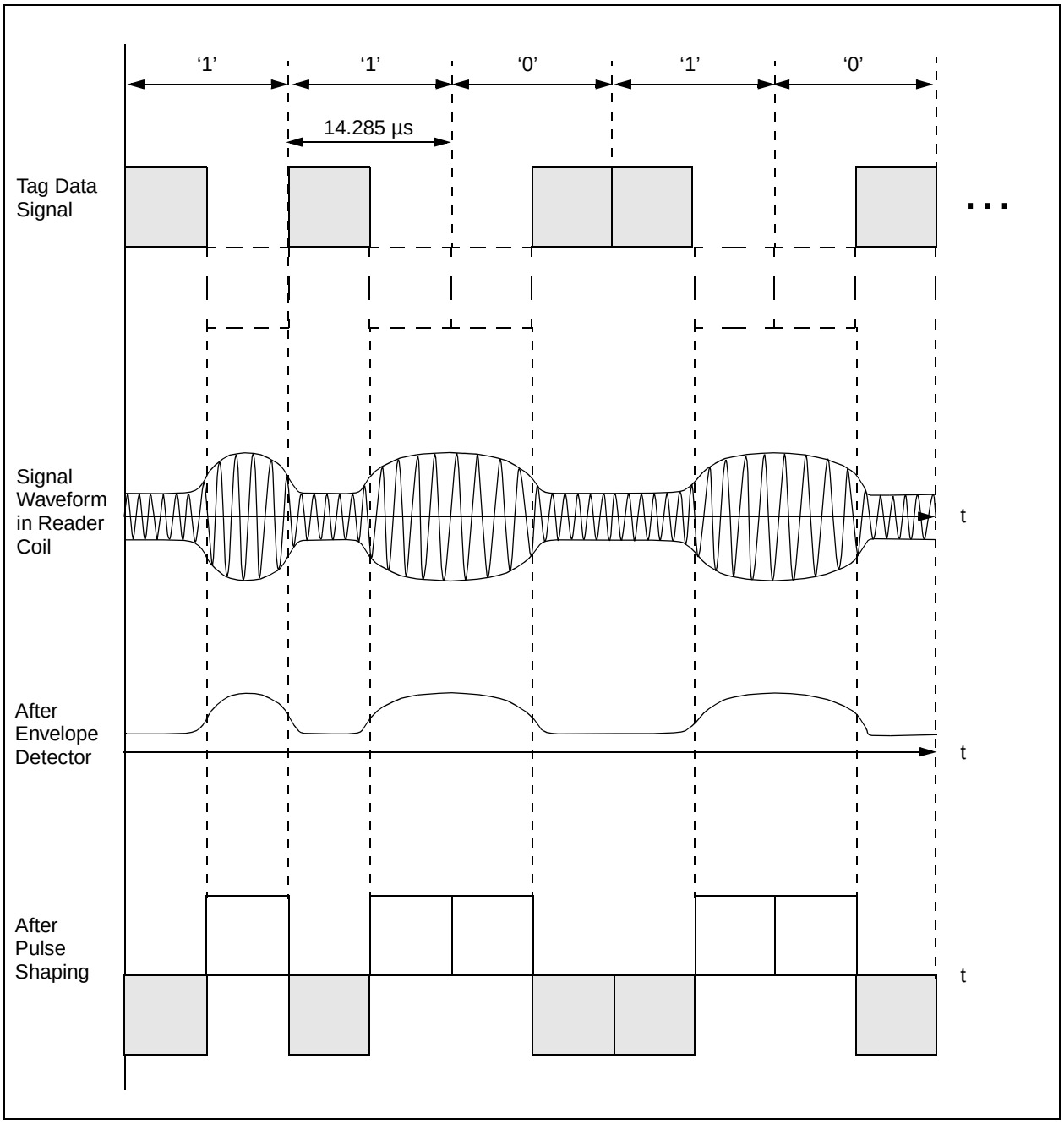
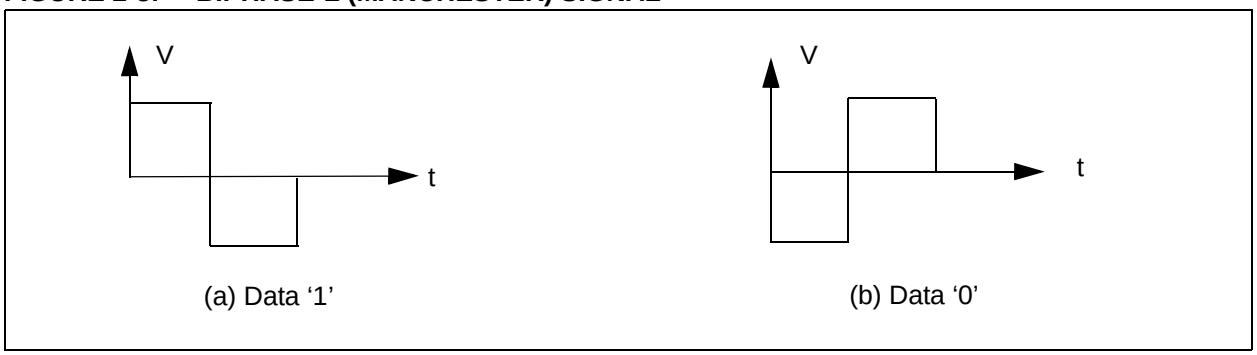


FIGURE 2-3: BIPHASE-L (MANCHESTER) SIGNAL



When the tag is energized by the reader's carrier signal, it transmits back with an amplitude modulated signal. This results in a perturbation in the voltage amplitude across the reader antenna coil. The envelope detector detects the changes in the voltage amplitude and passes it into an RC filter (R7, C11). The charged signal in the capacitor passes through active filters and amplifiers. The signal that is passing through this receiving section is the data signal. This filtered-shaped data signal is fed into Pin 10 of the microcontroller for data processing.

2.1 FCC Specifications on Transmitting Signal

Each country limits the signal strength of the radio frequency signal that is intentionally radiated from the device. In the USA, the maximum signal strength that is radiated from the device is regulated by Federal Communication Commission (FCC). Any device operating at 13.56 MHz frequency band must comply with the FCC Part 15.225 of the federal regulation. FCC limits for 13.56 MHz frequency band are al follows:

1. Tolerance of the carrier frequency: 13.56 MHz +/- 0.01% = +/- 7 kHz.
2. Frequency bandwidth: +/- 7 kHz.
3. Power level of fundamental frequency: 10 mv/m at 30 meters from the transmitter.
4. Power level for harmonics: -50.45 dB down from the fundamental signal.

The transmission circuit including the antenna coil must be designed to meet the FCC limits.

3.0 OPTIMIZATION FOR LONG-RANGE APPLICATIONS

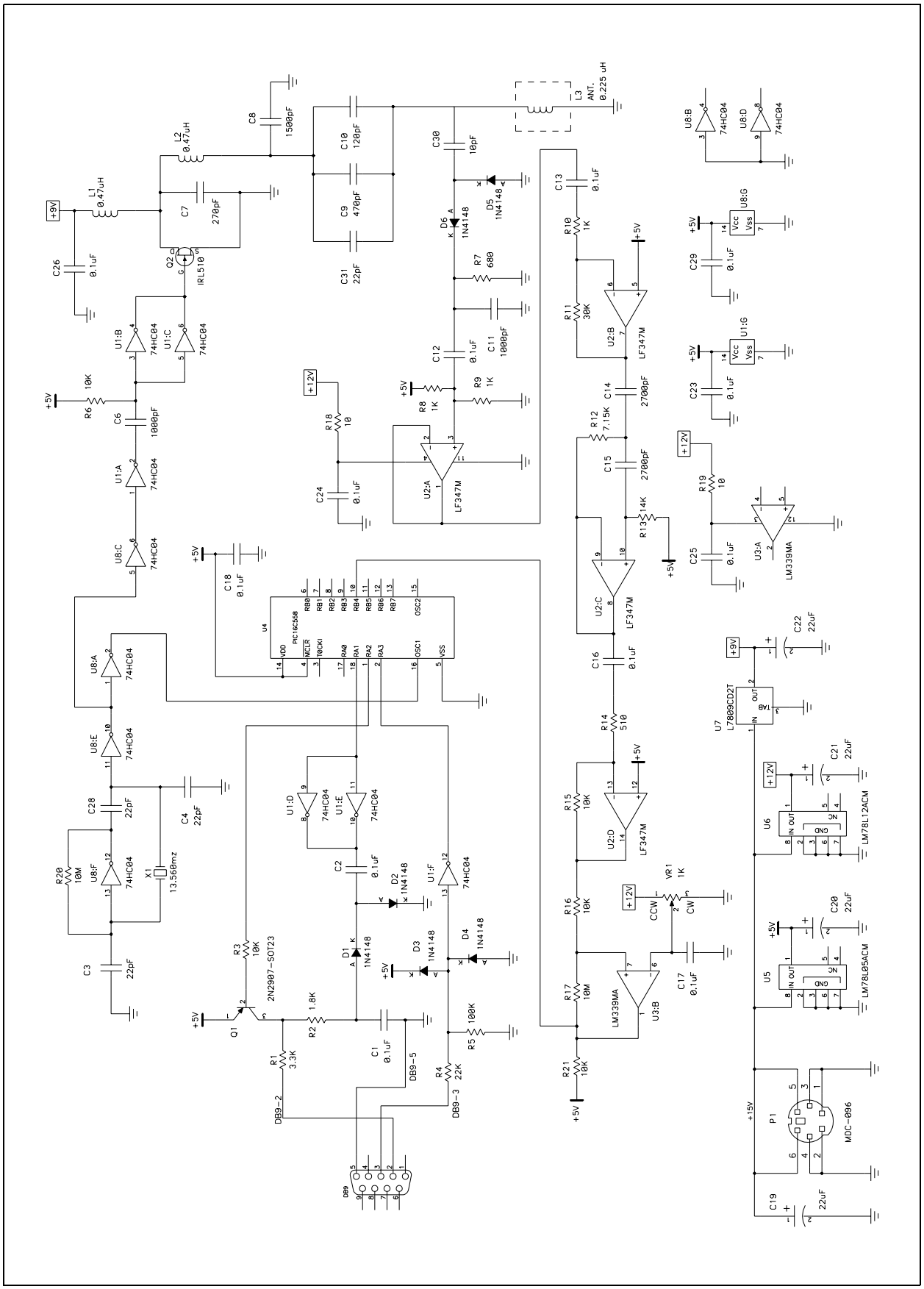
The reader circuit provided is designed for about a 5-inch read-range, using a 2-inch by 2-inch tag coil that is printed on PCB with the MCRF355. The read-range can be increased by increasing the reader power, sensitivity, and antenna size. A read-range of more than 30-inches can be achieved with the MCRF355 and an optimized reader. In order to optimize the reader circuit for long-range applications, the following aspects may be considered:

1. **Optimize the output power level within FCC limits.** The reader should provide a sufficient signal level to the tag. The tag needs about 4 V_{PP} across the coil circuit for operation. The power level radiating from the reader antenna must comply to the government regulations such as FCC specifications in the USA. The FCC limits for 13.56 MHz band are described in Section 2.1. For long-range applications, the designer may start with about 50 V_{PP} of antenna voltage and optimize the signal strength for a read-range within the government regulations.
2. **Increase the size of the antenna.** The read-range, in general, is proportional to the size of the reader coil (see Equation 12 in Application Note 710). An optimum radius of antenna is 1.414 times of the read-range.
3. **Increase the *Q* of the antenna circuit.** The read-range increases with Q of the antenna circuit. This is because the induced voltage is directly proportional to Q of the circuit. The recommended Q for long-range applications is as follows:

   $40 < Q < 96$ for reader
   $40 < Q$ for tag

4. **Optimize the input sensitivity of the reader.** The sensitivity is a measure of how weak a signal can be and still be satisfactorily received. The sensitivity is proportional to the carrier power and square of the modulation index (1 for 100% modulation such as MCRF355). It is inversely proportional to the noise signal. The limit to the sensitivity of the receiving section of the reader is noise, both external and internal. The external noises may come from various sources such as computers, televisions, appliances, motors, power lines, transformers, etc. The internal noise is mostly due to a thermal noise of components. To reduce noise, the reader should be operated a distance away from the noise sources. The receiving section may have a 70 kHz bandpass filter to reduce the noises. The 70 kHz bandpass filter will pass only the 70 kHz data signal for processing. The receiving section should have sensitivity of about -120 dBm for long-range applications.
5. **Optimize the amplitude gain circuit.** The receiving circuit amplifies the modulated signals before data processing. The input signal contains both real data and noise. Typically, op amplifiers are used for both as a gain amplifier and filter. The gain must be optimized within the circuit to obtain gains only at the real data signal.

4.0 READER SCHEMATIC



5.0 READER BILL OF MATERIALS

Assembly #	Line #	Qty	Part #	Part Description	Reference Designator
02-01523	1	1	02-01523-D	PCB ASSY DWG, MCRF355 microID READER	
02-01523	2	1	03-01523	SCHEMATIC, MCRF355 microID READER	
02-01523	3	1	04-01523	PCB FABRICATION, MCRF355 microID READER	
02-01523	4	2	MM74HC04M	IC, SMT, CMOS HEX INVERTER, 14P SOIC	U1, U8
02-01523	5	1	LF347M	IC, SMT, QUAD BI-FET OP AMP, 14P SOIC	U2
02-01523	6	1	LM339M	IC, SMT, LOW POWER LOW OFFSET VOLT QUAD COMPARATORS,14P SOIC	U3
02-01523	7	1	PIC16C558-20/SO	IC, PIC16C558-20/SO EPROM-BASED 8-BIT CMOS MICROCONTROLLER	U4
02-01523	8	1	LM78L05ACM	IC, REG, +5V 100 mA REGULATOR	U5
02-01523	9	1	LM78L12ACM	IC, REG, +12V 100 mA REGULATOR	U6
02-01523	10	1	L7809CD2T	IC, +9V, REG 1.5A TO-263	U7
02-01523	11	1	MMBT2907ALT1	TRANSISTOR, PNP, 2N2907A, SOT-23	Q1 Flip upside and bend legs toward the PCB
02-01523	12	1	IRL510	TRANSISTOR, N-CHANNEL HEX FET, TO220AB	Q2
02-01523	13	6	RLS4148TE11C	DIODE SMT, ROHM DIODE LL-34 SIG DIODE	D1-D6
02-01523	14	1	ERJ-3GSYJ332V	RES SMT, 3.3K OHM, 1/16W, 5%, 0603	R1
02-01523	15	1	ERJ-3GSYJ182V	RES SMT, 1.8K OHM, 1/16W, 5%, 0603	R2
02-01523	16	5	ERJ-3GSYJ103V	RES SMT, 10K OHM, 1/16 W, 5%, 0603	R3, R6, R15, R16, R21
02-01523	17	1	ERJ-3GSYJ223V	RES SMT, 22K OHM, 5% 0603	R4
02-01523	18	1	ERJ-3GSYJ104V	RES SMT, 100K OHM 1/16W 5% TYPE 0603	R5
02-01523	19	1	ERJ-3GSYJ681V	RES SMT, 680 OHM 1/16W 5% 0603	R7
02-01523	20	3	ERJ-3GSYJ102V	RES SMT, 1K OHM 1/16W 5% 0603	R8-R10
02-01523	21	1	ERJ-3GSYJ303V	RES SMT, 30K OHM 1/16W 5% 0603	R11
02-01523	22	1	ERJ-3EKF7151V	RES SMT, 7.15K OHM 1/16W 1% 0603	R12
02-01523	23	1	MFR-25FRF 14K0	RES, 14K OHM 1/4W 1% MF	R13, connected from U2 pin 12 to top pad of R13
02-01523	24	2	RM73B1JT106J	RES SMT, 10M OHM 1/16W 5% 0603	R17, R20
02-01523	25	2	ERJ-3GSYJ100V	RES SMT, 10 OHM 1/16W 5% 0603	R18, R19
02-01523	26	1	EVM-7JSX30B13	RES SMT, POT, 1K OHM 3MM SEALED, 3 TT	VR1
02-01523	27	12	ECU-V1H104KBW	CAP SMT, 0.1uF 50V 10%, X7R CER 1206	C1, C2, C12, C13, C16-18, C23-C26, C29
02-01523	28	3	ECU-V1H220JCV	CAP SMT, 22 pF CERAMIC 5% 50V 0603 NPO	C3, C4, C28
02-01523	29	2	ECU-V1H102KBV	CAP SMT, 1000 pF 50V CERAMIC 10% 0603 X7R	C6, C11
02-01523	30	1	ECU-V1H271JCV	CAP SMT, 270 pF 50V CERAMIC 5% 0603 NPO	C7
02-01523	31	1	ECU-V1H152KBV	CAP SMT, 1500 pF 50V CERAMIC 10% 0603 X7R	C8
02-01523	32	1	GRM42-6C0G471G500AL	CAP SMT, 470 pF 500V 2% 1206 C0G”	C9
02-01523	33	1	GRM42-6C0G121J500AL	CAP SMT, 120 pF 500V 5% 1206 C0G”	C10
02-01523	34	2	ECU-V1H272KBV	CAP SMT, 2700PF 50V CERAMIC 10% 0603 XR7	C14, C15
02-01523	35	4	ECE-A1EU220	CAP, 22UF 25V RADIAL ELECTROLYTIC 20%	C19-C22

Assembly #	Line #	Qty	Part #	Part Description	Reference Designator
02-01523	36	1	GRM42-6C0G100J500AL	CAP SMT, 10 pF 500V 5% 1206 C0G	C30
02-01523	37	1	GRM42-6C0G220J500AL	CAP SMT, 22 pF 500V 5% 1206 C0G	C31 (AS NEEDED)
02-01523	38	2	43LS477	INDUCTOR, 0.47 µH	L1, L2
02-01523	39	1	MCX0001	OSCILLATOR, CUSTOM 13.560 MHz, PARALLEL MODE, 22 pF LOAD, HC49 CASE, 30 PPM	X1
02-01523	40	1	MDC-096	CONN, MINI-DIN, 6-PIN	P1
02-01523	41	1	KF22-E9S-NJ	CONN, D-SUB 9P RECPT RT ANGLE WITH JACK SCREWS	DB9
02-01523	42	1	08-00170	LABEL, MCRF355 READER FIRMWARE, 355READ.HEX, 1/25/99, U4	@ U4
02-01523	43	1	ERJ-3GSYJ511V	RES SMT, 510 OHM 1/16W 5% 0603	R14

6.0 READER SOURCE CODE FOR THE PICmicro® MCU

```
;receiver.asm



;Processor: PIC16C558 operating at 13.56 MHz
;          Ti= 295 nsec

        processor 16c558
        #include “P16c558.inc”
        __config h’3ff2’  ;protection off,PWRT enabled,watchdog disabled,HS oscillator


#define _CARRY          STATUS,0
#define _ZERO           STATUS,2

#define _125KHZ         PORTA,1
#define _RS232TX        PORTA,2
#define _RS232RX        PORTA,3
#define _RS232          PORTA
#define SIGNAL          PORTB,4
invmask               = h’2’
;.............................................................
;Define variables and constants here--
delay       =h’20’
wait        =h’21’
acctime     =h’22’ ;accumulated sync interval sum--also used as halfbit interval threshold
#define     halfthr     acctime  ;halfbit interval threshold
halfthr     =acctime ;halfbit interval threshold
recv_csumhi =h’23’ ;2 bytes for storing received checksum
recv_csumlo =h’24’
bitcnt      =h’25’ ;RS232 bit counter
cycle_cnt   =h’26’
halfthr     =h’27’ ;threshold value between halfbit and fullbit intervals
ptr1        =h’28’ ;temporary FSR storage
ptr2        =h’29’ ;temporary FSR storage
TXchar      =h’2a’ ;character to transmit over RS232
temp        =h’2b’ ;temporary storage
shiftcnt    =h’2c’ ;used to strip the framing ‘0’ bits from the rec’d data array
letters    =h’2d’ ;storage area for next character to send
charcnt     =h’2e’
lastbit     =h’2f’ ;the LSb stores the last rec’d bit--flip it by complementing f

;;;!!!!!!!!!!!!!!!!!!!!bit storage area--16 bytes of storage, indirectly addressed
;;;Note that s/w tests for MSb to detect end of area--be careful if move to different
;;;processor or relocate this storage area
recvbits    =h’40’  ;32 bytes set aside for storing the received bits--actual number of bytes
                    ;in transmission is 18
;;Note that main loop uses bit tests to determine bit receive or runaway condition (to limit
;;processing time). Keep this in mind if recvbits storage area changed in the future.
;;40h-60h is reserved for received bits--actual bit receiving area 40h-51h, rest is overrun area

;;52h-73h set aside for ASCII conversion of received bytes before RS232 transmission. Note that
;;52h-60h contains no useful information from the use during receive of demodulated bits. Also,
;;        bits are not being received while the ASCII conversion and serial transmission are
;;        taking place.
;;        ‘G’ 1st character: “go”
;;        Character 2-37:    ASCII representation of received 18 bytes (until checksum used)
;;        Character 38:      ‘\n’ newline

sendascii   =h’52’  ;begin of storage area for ASCII conversion of received bytes
xfercnt     =d’14’  ;defines number of received bytes to convert to ASCII & transmit

;.....................................................................
```

```
;.....................................................................................
;Overall function- To recover Manchester encoded RFID message after AM demodulation and
;                  comparator decision. The comparator input trips the interrupt on PORTB change.
;The steps are:
;
;     1- Initialize registers to seek synch field.
;     2- Determine bit width from synch field by averaging the periods between transitions
;        over the synch field. TMR0 is cleared at each edge. If the timer overflows before
;          the next edge, synch seek starts over. The synch field is composed of 9 bits.
;     3- Use the measured bit width to establish a threshold period between repeat bits and
;          complement of previous bit. This is due to the Manchester encoding method. Since there
;          is always a transition in the middle of each bit interval transmitted, a repeated bit
;          will appear as a pair of edges that occur with a halfbit interval period. A bit that
;          is the complement of the last received bit will appear as an interval between edges
;          of a full bit interval period.
;     4- Shift in bits as they are received into the storage array. When the timer overflows,
;          consider the data field over. The received data format is MSb to LSb, where the MSb
;          is the first bit received.
;     5- There are 16 bytes in the message, followed by a 16 bit checksum of the message
;          contents. The remaining bit is unused.
;       6- Compute the checksum of the received 16 byte message and compare to the received
;          checksum.
;       7- If checksums match, convert the message and the checksum into ASCII form and transmit
;          over the RS232 serial link. The message format is:
;                 “GG” :the go characters (start of message)
;                 36 bytes which are the ASCII representation of the 18 bytes received
;                 “\n” : closing newline character
;          The serial data rate is 9600 bps, 8 data bits, 1 stop, no parity
;.....................................................................


        org h’000’           ;RESET vector location
        goto   init
        org h’004’           ;interrupt vector location

;=======================================================================================
;;isr(): interrupt service routine
;        interrupts enabled for transition on PORTB
;
;        1- BEWARE! To minimize interrupt response time, the w & status register are NOT
;           archived.
;      2- The isr execution path is determined by w register and uses calculated goto’s.
;           The w for next isr is set at end of current isr execution and is dependent on
;           signal context (i.e. sync start, w/in sync, w/in data, etc.)
;         Be very cautious here--must stay w/in 255 instructions for this to work!
;      3- Sync field processed as follows:
;             -Ignore the first 4 transitions, they may be in response to tag power on reset
;             -Accumulate the sum of next 8 intervals
;             -Establish half bit width from full bit width threshold value based on
;              average interval measured above. Due to Manchester encoding, repeat of previous
;              bit will be a series of 2 halfbit width intervals, complement of previous bit
;              will be a fullbit width interval. halfbit defined as 1.5x(average sync).
;                -wait for interval over the fullbit threshold. This is end of sync. In accordance
;              w/ Manchester encoding, the sync field will be: 1 1 1 1 1 1 1 1 0
;=======================================================================================
isr
      addwf  PCL,f       ;4 calculated goto
;first sync edge is calculated goto here
      clrf   TMR0        ;5
      movf   PORTB,f     ;6 must read PORTB before clearing RBIF
      bcf    INTCON,RBIF ;7 just in case timer interrupt happened just at 1st edge
      bcf    INTCON,T0IF ;8
      movlw  (first_cycle - isr-d’1’) ;9 next isr calculated goto offset
      clrf   lastbit     ;10  lastbit @ end of sync = 0
      retfie             ;12
;end of first cycle here. Note that first 4 transitions are ignored, because sync start is
```

```
;corrupted by tag power on reset.
first_cycle
      clrf   TMR0        ;5
      movf   PORTB,f     ;6 must read PORTB before clearing RBIF
      bcf    INTCON,RBIF ;7
      movlw  (second_cycle - isr-d'1') ;8 next isr calculated goto offset
      retfie             ;10
;end of 2nd cycle here. Note that first 4 transitions are ignored, because sync start is
;corrupted by tag power on reset.
second_cycle
      clrf   TMR0        ;5
      movf   PORTB,f     ;6 must read PORTB before clearing RBIF
      bcf    INTCON,RBIF ;7
      movlw  recvbits    ;8
      movwf  FSR         ;9 set up to store data bits
      movlw  (third_cycle - isr-d'1') ;10 next isr calculated goto offset
      retfie             ;12
;end of 3rd cycle here. Note that first 4 transitions are ignored, because sync start is
;corrupted by tag power on reset. The 3rd cycle is the 4th transition, so from here we measure
;the longest interval in sync field.
third_cycle
      clrf   TMR0        ;5
      movf   PORTB,f     ;6 must read PORTB before clearing RBIF
      bcf    INTCON,RBIF ;7
      clrf   acctime     ;8  reset accumulated sync interval for average
      movlw  (fourth_cycle - isr-d'1') ;9 next isr calculated goto offset
      retfie             ;11
;end of 4th cycle here. Start looking for longest sync interval here.
fourth_cycle
      movf   TMR0,w      ;5
      clrf   TMR0        ;6
      movf   PORTB,f     ;7
      bcf    INTCON,RBIF ;8
      addwf  acctime,f   ;9  first measured sync cycle, must be the largest
      movlw  (fifth_cycle - isr-d'1') ;10
      retfie             ;12
;end of 5th cycle here.
fifth_cycle
      movf   TMR0,w      ;5
      clrf   TMR0        ;6
      movf   PORTB,f     ;7
      bcf    INTCON,RBIF ;8
      addwf  acctime,f   ;9   acctime = acctime + TMR0
      movlw  (sixth_cycle - isr-d'1') ;10
      retfie             ;12
;end of 6th cycle here.
sixth_cycle
      movf   TMR0,w      ;5
      clrf   TMR0        ;6
      movf   PORTB,f     ;7
      bcf    INTCON,RBIF ;8
      addwf  acctime,f   ;9   acctime = acctime + TMR0
      movlw  (seventh_cycle - isr-d'1') ;10
      retfie             ;12
;end of 7th cycle here.
seventh_cycle
      movf   TMR0,w      ;5
      clrf   TMR0        ;6
      movf   PORTB,f     ;7
      bcf    INTCON,RBIF ;8
      addwf  acctime,f   ;9   acctime = acctime + TMR0
      movlw  (eighth_cycle - isr-d'1') ;10
      retfie             ;12
;end of 8th cycle here.
eighth_cycle
      movf   TMR0,w      ;5
```

```
      clrf   TMR0        ;6
      movf   PORTB,f     ;7
      bcf    INTCON,RBIF ;8
      addwf  acctime,f   ;9   acctime = acctime + TMR0
      movlw  (nineth_cycle - isr-d'1') ;10
      retfie             ;12
;end of 9th cycle here.
nineth_cycle
      movf   TMR0,w      ;5
      clrf   TMR0        ;6
      movf   PORTB,f     ;7
      bcf    INTCON,RBIF ;8
      addwf  acctime,f   ;9   acctime = acctime + TMR0
      movlw  (tenth_cycle - isr-d'1') ;10
      retfie             ;12
;end of 10th cycle here.
tenth_cycle
      movf   TMR0,w      ;5
      clrf   TMR0        ;6
      movf   PORTB,f     ;7
      bcf    INTCON,RBIF ;8
      addwf  acctime,f   ;9   acctime = acctime + TMR0
      movlw  (eleventh_cycle - isr-d'1') ;10
      retfie             ;12
;end of 11th cycle here. --this is last of sync cycles to be accumulated. Average the result
;and determine halfbit threshold in remaining sync cycles.
eleventh_cycle
      movf   TMR0,w      ;5
      clrf   TMR0        ;6
      movf   PORTB,f     ;7
      bcf    INTCON,RBIF ;8
      addwf  acctime,f   ;9   acctime = acctime + TMR0
      movlw  (twelfth_cycle - isr-d'1') ;10
      retfie             ;12
;end of 12th cycle here. Start averaging the sync interval accumulated time
twelfth_cycle
      movf   PORTB,f     ;5
      bcf    INTCON,RBIF ;6
      rrf    acctime,f   ;7 acctime/2
      rrf    acctime,f   ;8 acctime/4
      rrf    acctime,f   ;9 avg interval = acctime/8
      movlw  h'1f'       ;10   clear 3 MSbs that may have been set by carry
      andwf  acctime,f   ;11
      movlw  (cycle13 - isr-d'1') ;12
      retfie             ;14
;end of 13th cycle here. Calculate the halfbit threshold = 1.5(sync interval avg) Note that
;that the threshold value will be kept in acctime (=halfthr)
cycle13
      clrf   TMR0        ;5
      movf   PORTB,f     ;6
      bcf    INTCON,RBIF ;7
      rrf    acctime,w   ;8 half the sync interval avg
      addwf  acctime,f   ;9 halfthr = 1+1.5x(sync interval avg)
      incf   acctime,f   ;10
      movlw  (sync_end - h'100'-h'1'-isr) ;11
      bsf    PCLATH,0    ;12  adjust for origin @ 100h
      retfie             ;14

      org   h'100'
;sync end wait. End of sync is distinguished by a fullbit interval. ( T > halfthr )
sync_end
      movf  TMR0,w       ;5
      clrf  TMR0         ;6
      movf  PORTB,f      ;7
      bcf   INTCON,RBIF  ;8
      subwf halfthr,w    ;9    Test interval to detect end of sync field (halfthr - w)
```

```
      movlw (sync_end - h'100'-isr-d'1') ;10
      btfss STATUS,C     ;12   Carry set for halfthr >= w
      movlw (bit1 - h'100'-isr-h'1');12   If T > halfbit, end of sync detected. Proceed to data
processing
      retfie             ;14
;rec'd bit processing here --bit1 is 1st bit of 8 bit block
bit1
      movf  TMR0,w       ;5
      clrf  TMR0         ;6
      movf  PORTB,f      ;7
      bcf   INTCON,RBIF  ;8
      subwf halfthr,w    ;9   Test interval to determine bit. C = 1 for repeated bit
      btfsc STATUS,C     ;11
      goto  halfabit1    ;12
;fullbit processing here
      comf  lastbit,f    ;12  Complement lastbit for fullbit measurement
      rrf   lastbit,w    ;13
      rlf   INDF,f       ;14  shift in the new bit
      movlw (bit2 - h'100'-isr-h'1') ;15
      retfie             ;17
halfabit1
;repeated bit (1 of 8)
      rrf   lastbit,w    ;13
      rlf   INDF,f       ;14
      movlw (half21-h'100'-isr-h'1') ;15
      retfie             ;17
;2nd half of bit interval processing
half21       ;2nd half, bit1
      clrf  TMR0         ;5
      movf  PORTB,f      ;6
      bcf   INTCON,RBIF  ;7
      movlw (bit2-h'100'-isr-h'1');8
      retfie             ;10

;rec'd bit processing here --bit2 is 2nd bit of 8 bit block
bit2
      movf  TMR0,w       ;5
      clrf  TMR0         ;6
      movf  PORTB,f      ;7
      bcf   INTCON,RBIF  ;8
      subwf halfthr,w    ;9   Test interval to determine bit. C = 1 for repeated bit
      btfsc STATUS,C     ;11
      goto  halfabit2    ;12
;fullbit processing here
      comf  lastbit,f    ;12  Complement lastbit for fullbit measurement
      rrf   lastbit,w    ;13
      rlf   INDF,f       ;14  shift in the new bit
      movlw (bit3 - h'100'-isr-h'1') ;15
      retfie             ;17
halfabit2
;repeated bit (2 of 8)
      rrf   lastbit,w    ;13
      rlf   INDF,f       ;14
      movlw (half22-h'100'-isr-h'1') ;15
      retfie             ;17
;2nd half of bit interval processing
half22       ;2nd half, bit2
      clrf  TMR0         ;5
      movf  PORTB,f      ;6
      bcf   INTCON,RBIF  ;7
      movlw (bit3-h'100'-isr-h'1');8
      retfie             ;10
;rec'd bit processing here --bit3 is 3rd bit of 8 bit block
bit3
      movf  TMR0,w       ;5
      clrf  TMR0         ;6
```

```
      movf  PORTB,f      ;7
      bcf   INTCON,RBIF  ;8
      subwf halfthr,w    ;9   Test interval to determine bit. C = 1 for repeated bit
      btfsc STATUS,C     ;11
      goto  halfabit3    ;12
;fullbit processing here
      comf  lastbit,f    ;12  Complement lastbit for fullbit measurement
      rrf   lastbit,w    ;13
      rlf   INDF,f       ;14  shift in the new bit
      movlw (bit4 - h'100'-isr-h'1') ;15
      retfie             ;17
halfabit3
;repeated bit (3 of 8)
      rrf   lastbit,w    ;13
      rlf   INDF,f       ;14
      movlw (half23-h'100'-isr-h'1') ;15
      retfie             ;17
;2nd half of bit interval processing
half23       ;2nd half, bit3
      clrf  TMR0         ;5
      movf  PORTB,f      ;6
      bcf   INTCON,RBIF  ;7
      movlw (bit4-h'100'-isr-h'1');8
      retfie             ;10
;rec'd bit processing here --bit4 is 4th bit of 8 bit block
bit4
      movf  TMR0,w       ;5
      clrf  TMR0         ;6
      movf  PORTB,f      ;7
      bcf   INTCON,RBIF  ;8
      subwf halfthr,w    ;9   Test interval to determine bit. C = 1 for repeated bit
      btfsc STATUS,C     ;11
      goto  halfabit4    ;12
;fullbit processing here
      comf  lastbit,f    ;12  Complement lastbit for fullbit measurement
      rrf   lastbit,w    ;13
      rlf   INDF,f       ;14  shift in the new bit
      movlw (bit5 - h'100'-isr-h'1') ;15
      retfie             ;17
halfabit4
;repeated bit (4 of 8)
      rrf   lastbit,w    ;13
      rlf   INDF,f       ;14
      movlw (half24-h'100'-isr-h'1') ;15
      retfie             ;17
;2nd half of bit interval processing
half24       ;2nd half, bit4
      clrf  TMR0         ;5
      movf  PORTB,f      ;6
      bcf   INTCON,RBIF  ;7
      movlw (bit5-h'100'-isr-h'1');8
      retfie             ;10
;rec'd bit processing here --bit5 is 5th bit of 8 bit block
bit5
      movf  TMR0,w       ;5
      clrf  TMR0         ;6
      movf  PORTB,f      ;7
      bcf   INTCON,RBIF  ;8
      subwf halfthr,w    ;9   Test interval to determine bit. C = 1 for repeated bit
      btfsc STATUS,C     ;11
      goto  halfabit5    ;12
;fullbit processing here
      comf  lastbit,f    ;12  Complement lastbit for fullbit measurement
      rrf   lastbit,w    ;13
      rlf   INDF,f       ;14  shift in the new bit
      movlw (bit6 - h'100'-isr-h'1') ;15
```

```
      retfie            ;17
halfabit5
;repeated bit (5 of 8)
      rrf   lastbit,w    ;13
      rlf   INDF,f       ;14
      movlw (half25-h'100'-isr-h'1') ;15
      retfie            ;17
;2nd half of bit interval processing
half25       ;2nd half, bit5
      clrf  TMR0         ;5
      movf  PORTB,f      ;6
      bcf   INTCON,RBIF  ;7
      movlw (bit6-h'100'-isr-h'1')   ;8
      retfie            ;10

;rec'd bit processing here --bit6 is 6th bit of 8 bit block
bit6
      movf  TMR0,w       ;5
      clrf  TMR0         ;6
      movf  PORTB,f      ;7
      bcf   INTCON,RBIF  ;8
      subwf halfthr,w    ;9   Test interval to determine bit. C = 1 for repeated bit
      btfsc STATUS,C     ;11
      goto  halfabit6    ;12
;fullbit processing here
      comf  lastbit,f    ;12  Complement lastbit for fullbit measurement
      rrf   lastbit,w    ;13
      rlf   INDF,f       ;14  shift in the new bit
      movlw (bit7 - h'100'-isr-h'1') ;15
      retfie            ;17
halfabit6
;repeated bit (6 of 8)
      rrf   lastbit,w    ;13
      rlf   INDF,f       ;14
      movlw (half26-h'100'-isr-h'1') ;15
      retfie            ;17
;2nd half of bit interval processing
half26       ;2nd half, bit6
      clrf  TMR0         ;5
      movf  PORTB,f      ;6
      bcf   INTCON,RBIF  ;7
      movlw (bit7-h'100'-isr-h'1') ;8
      retfie            ;10
;rec'd bit processing here --bit7 is 7th bit of 8 bit block
bit7
      movf  TMR0,w       ;5
      clrf  TMR0         ;6
      movf  PORTB,f      ;7
      bcf   INTCON,RBIF  ;8
      subwf halfthr,w    ;9   Test interval to determine bit. C = 1 for repeated bit
      btfsc STATUS,C     ;11
      goto  halfabit7    ;12
;fullbit processing here
      comf  lastbit,f    ;12  Complement lastbit for fullbit measurement
      rrf   lastbit,w    ;13
      rlf   INDF,f       ;14  shift in the new bit
      movlw (bit8 - h'100'-isr-h'1') ;15
      retfie            ;17
halfabit7
;repeated bit (7 of 8)
      rrf   lastbit,w    ;13
      rlf   INDF,f       ;14
      movlw (half27-h'100'-isr-h'1') ;15
      retfie            ;17
;2nd half of bit interval processing
half27       ;2nd half, bit7
```

```
      clrf  TMR0         ;5
      movf  PORTB,f      ;6
      bcf   INTCON,RBIF  ;7
      movlw (bit8-h'100'-isr-h'1') ;8
      retfie             ;10
;rec'd bit processing here --bit8 is 8th bit of 8 bit block
bit8
      movf  TMR0,w       ;5
      clrf  TMR0         ;6
      movf  PORTB,f      ;7
      bcf   INTCON,RBIF  ;8
      subwf halfthr,w    ;9   Test interval to determine bit. C = 1 for repeated bit
      btfsc STATUS,C     ;11
      goto  halfabit8    ;12
;fullbit processing here
      comf  lastbit,f    ;12  Complement lastbit for fullbit measurement
      rrf   lastbit,w    ;13
      rlf   INDF,f       ;14  shift in the new bit
      movlw (bit1 - h'100'-isr-h'1') ;15
      incf  FSR,f        ;16
      retfie             ;18
halfabit8
;repeated bit (8 of 8)
      rrf   lastbit,w    ;13
      rlf   INDF,f       ;14
      movlw (half28-h'100'-isr-h'1') ;15
      retfie             ;17
;2nd half of bit interval processing
half28       ;2nd half, bit8
      clrf  TMR0         ;5
      movf  PORTB,f      ;6
      bcf   INTCON,RBIF  ;7
      movlw (bit1-h'100'-isr-h'1') ;8
      incf  FSR,f        ;9   advance to next byte in recvbits storage array
      retfie             ;11



;The negative RS232 supply is generated by an inverter clocked at ~125 KHz by port pin RA1.
        ;first pump up the -5V, i.e. generate 125 KHz clock (T=8 usec, ~27 Ti)
        ;run for a total of 128 cycles before sending data
       ;put line at stop bit level

alphabet
        clrwdt
      bcf   INTCON,GIE  ;make sure interrupts are off
      movlw  sendascii
      movwf  FSR

      movlw  xfercnt    ;# of ASCII represented received bytes to xfer
      addlw  xfercnt    ;x2
      addlw  h'3'       ;plus 2 start character "G" and newline character at end
      movwf  charcnt
;;set up registers in bank 1
        bsf    STATUS,RP0  ;point to bank 1
        movlw  h'8'
        movwf  TRISA       ;RA3 input, RA2-0 output
        movlw  h'10'
        movwf  TRISB       ;RB7-5,3-0 output, RB4 input
        movlw  b'00001100' ;set up timer option for internal clock, prescale-->watchdog/16
        movwf  OPTION_REG  ;port B pullups enabled
        bcf    STATUS,RP0  ;point back to bank 0
;;done setting up registers in bank 1, back to bank 0
        bsf    _RS232TX  ;default is mark mode
        call   gen125khz
```

```
;start the test transmission
sendA
        movf   INDF,w
        movwf  TXchar
        movlw  d'8'
        movwf  bitcnt
       ;stop bit last
        bsf    _RS232TX
        call   TX_RS232    ;stop bit = 3Ti


        call   ti17        ;burn 17Ti (includes the 2Ti for the call)
        ;start bit first
        bcf    _RS232TX
        call   TX_RS232
        call   ti17       ;burn 17Ti (includes the 2Ti for the call, adjusts the bit timing)
sendchar
        btfsc  TXchar,0    ;1Ti
        goto   setbit      ;3Ti
        bcf    _RS232TX
        goto   nextbit
setbit
        bsf    _RS232TX    ;4Ti
nextbit
        call   TX_RS232    ;6Ti
        rrf    TXchar,f    ;7Ti
        call   ti10        ;17Ti
        decfsz bitcnt,f    ;18Ti
        goto   sendchar    ;20Ti
       ;stop bit last
        bsf    _RS232TX
        call   TX_RS232    ;stop bit = 3Ti


     incf   FSR,f      ;1
     decfsz charcnt,f  ;2
     goto   inalpha    ;4
     movlw  d'255'
     movwf  charcnt
     movlw  d'10'
     movwf  bitcnt
waiting
     call   ti17
     decfsz charcnt,f
     goto   waiting
     decfsz bitcnt,f
     goto   waiting
     goto   seekinit
inalpha
     call   ti10
     goto   sendA


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;;subroutine--RS232 bit timing & 125 KHz voltage inverter maintenance
;;            baud rate set to 9600 bps--this is a bit time of 104 usec
;;            Timing for this subroutine: to104 loop is 5.605 usec, additional setup
;;                                        overhead is 1.77 usec. If do 17 to104 loops,
;;                                        that leaves 5.844 usec to make up in the calling
;;                                        routine to meet 104 usec target. 5.844= 19.8 Ti
;;                                        (20 Ti)
;;                                        Note that 5.844 is not evenly divisible by the
;;                                        instruction cycle time. Need to save one
;;                                        instruction every 5th bit sent--w/ the stop & start
;;                                        bit overhead, easier to save 2 extra instructions
```

```
;;                                        every character sent (10 bits)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
TX_RS232
        movlw  d'17'    ;time out 104 usec, Ti=295 nsec
        movwf  wait
to104
        movlw  invmask   ;flip voltage inverter bit
        xorwf  _RS232,f
        movlw  d'4'
        movwf  delay
wait4usec
        decfsz delay,f     ;4 usec is half inverter clock period
        goto   wait4usec
        decfsz wait,f
        goto   to104
        movlw  invmask
        xorwf  _RS232,f
        nop
        nop
        nop
        return


;;==================================================================================


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;;subroutine--generates 128 cycles at ~125 KHz for the RS232 voltage inverter
gen125khz
        movlw  d'128'
        movwf  cycle_cnt
next125
        bsf    _125KHZ
        movlw  d'4'
        movwf  delay
highside
        decfsz delay,f
        goto   highside
        bcf    _125KHZ
        movlw  d'4'
        movwf  delay
lowside
        decfsz delay,f
        goto   lowside
        decfsz cycle_cnt,f
        goto   next125
        return
;;end gen125khz
subroutine;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;subroutine-ti17: burn 17 Ti--includes the 2Ti to call this subroutine
;;           ti15: burn 15 Ti, including call
;;           ti10: burn 10 Ti, including call
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ti17
        movlw  d'3'     ;1
        movwf  delay    ;2
burn9
        decfsz delay,f
        goto   burn9    ;11
        clrwdt          ;12
```



```
        nop             ;13
        return          ;15+2 for call ti17=17Ti

ti15
        movlw  d'3'     ;1
        movwf  delay    ;2
burn9Ti
        decfsz delay,f
        goto   burn9Ti  ;11
        return          ;13+2 for call ti15=15Ti

ti12
      nop
      clrwdt
ti10
        goto   dly1      ;2Ti
dly1
        goto   dly2      ;4Ti
dly2
        goto   leaveti10 ;6Ti
leaveti10
        return           ;8Ti+2Ti=10Ti



;=================
;;initialization
;=================
init
        ;1st set up the I/O configuration--note that setting PORTB 7,6,5,4,0 as outputs disables
        ;them as external interrupt sources. In this application PORTB-4 is utilized as an
        ;external interrupt source upon change of state. All other external interrupt sources are
        ;set as outputs to disable them as interrupts.

;;set up registers in bank 1
        bsf    STATUS,RP0  ;point to bank 1
        movlw  h'8'
        movwf  TRISA       ;RA3 input, RA2-0 output
        movlw  h'10'
        movwf  TRISB       ;RB7-5,3-0 output, RB4 input
        movlw  b'00001000' ;set up timer option for internal clock, no prescaler
        movwf  OPTION_REG  ;port B pullups enabled
        bcf    STATUS,RP0  ;point back to bank 0
;;done setting up registers in bank 1, back to bank 0
      movlw  HIGH isr
      movwf  PCLATH      ;setup for calculated goto's dependent on context when entering
                            ;isr
;=====================================================
;;initialization for sync field search- done @ turn on & after data recovery complete (or failed)
;=====================================================
seekinit
        clrwdt
      movlw  d'19'
      movwf  bitcnt    ;clear the bit storage field
      movlw  recvbits
      movwf  FSR
clrbits
      clrf   INDF
      incf   FSR,f
      decfsz bitcnt,f
      goto   clrbits


        movlw  recvbits
        movwf  FSR        ;start of the received bits field
      movf   PORTB,w ;read PORTB before clearing INTCON to be sure RBIF=0
```

```
      clrf   INTCON
      clrf   TMR0
;*******************************************************************************
; From here on, the w register represents the PCL offset when answering the isr.
; It is to be used for no other purpose until interrupts are disabled.
;*******************************************************************************
      movlw  d'0'
      clrf   PCLATH
      bsf    INTCON,RBIE  ;enable portB change interrupt enable
      bsf    INTCON,GIE   ;global interrupts are now enabled.


;====================
;;tag word search
;====================
;The main loop monitors the T0IF flag to detect successfully received word (subject to
;checksum test). Tag word processing is isr driven. A calculated goto method is used for
;position context in tag word for speed. FOR THIS REASON, THE W REGISTER CANNOT BE USED
;BY THE MAIN LOOP! If the main loop detects a timer overflow, the w register is cleared to
;return processing to first sync edge search.
;Also, expect recvbits area to be @ 40h-52h while receiving data. The ptr will be tested to
;determine this bitwise (because w can't be used in the main loop).
;*********************************************************************************************
seeksync
      bcf     INTCON,RBIE
      movlw   d'0'    ;calculated goto offset for 1st sync edge processing
      clrf    PCLATH
      clrf    FSR     ;FSR = 0 to indicate not gathering bits
      bsf     INTCON,RBIE
      bcf     INTCON,T0IF
main
        clrwdt
      btfsc   FSR,6
      goto    datamain   ;receiving data, monitor progress
      btfsc   INTCON,T0IF
      goto    seeksync   ;if TMR0 overflows w/o receiving bits, seeksync
        goto    main
;check for done receiving bits using TMR0 overflow as indicator. Also test for overflow from
;proper bit storage area for runaway condition (non tag noise tripping comparator)
datamain
      clrwdt
      btfsc   INTCON,T0IF
      goto    calc_checksum ;if timer overflows, calculate checksum of received data
      btfsc   FSR,5        ;if bit 5 set, FSR > 5fh and has overrun its proper area.
      goto    seeksync     ;search for sync.
      goto    datamain
;Data received at this point. Two processing tasks remain:
;1- the framing '0' bits must be removed from the received 14 data bytes and 16 bit checksum
;2- the checksum of the 14 data bytes must be calculated and compared to the received
;   16 bit checksum
;If checksums match, transmit data over RS232 link.

calc_checksum
      clrf    INTCON
clrgie
      bcf     INTCON,GIE
        btfsc   INTCON,GIE   ;make sure it's clear before proceeding
      goto    clrgie
      movf    PORTB,f
      clrf    INTCON       ;disable all interrupts while processing received data
;remove the framing '0' bits by bit shifting the data array left until all framing 0s are
;shifted out

      movlw   d'17'
      movwf   bitcnt
      movwf   shiftcnt
```

```
shiftout
      movlw   recvbits+d'17'
      movwf   FSR
roll_left
      rlf     INDF,f
      decf    FSR,f
      decfsz  shiftcnt,f
      goto    roll_left     ;rotate left shiftcnt # of bytes
      decfsz  bitcnt,f
      goto    next_RL
      goto    framestripped
;bit shift left through the array (successively 1 byte less each time)
next_RL
      movf    bitcnt,w
      movwf   shiftcnt
      goto    shiftout
framestripped

;1st check for all 0s in data--This is an illegal combination
      movlw   recvbits
      movwf   FSR
      movlw   d'14'
      movwf   bitcnt
zerotest
      movf    INDF,w
      btfss   STATUS,Z
      goto    nonzero
      decfsz  bitcnt,f
      goto    zerotest
      goto    seekinit     ;all zeros received. Ignore the message
nonzero
;do 16 bit checksum of first 14 bytes received. It should match the last 2 bytes received.
      movlw   recvbits
      movwf   FSR
      movlw   d'14'
      movwf   bitcnt
      clrf    recv_csumlo
      clrf    recv_csumhi
sumbytes
      movf    INDF,w
      addwf   recv_csumlo,f
      btfsc   STATUS,C
      incf    recv_csumhi,f  ;carry into high byte as necessary
      incf    FSR,f          ;point to next data byte
      decfsz  bitcnt,f
      goto    sumbytes
;now compare the received checksum w/ the calculated checksum. Transmit data if they match.
      movf    recv_csumhi,w
      subwf   INDF,f
      btfss   STATUS,Z
      goto    seekinit
      incf    FSR,f         ;point to received checksum LSB
      movf    recv_csumlo,w
      subwf   INDF,f
      btfss   STATUS,Z
      goto    seekinit
;message passes checksum. Convert to ASCII and transmit.
;now convert to ASCII form
      movlw    recvbits
      movwf    ptr1       ;keep track of where in conversion
      movlw    sendascii
      movwf    ptr2
      movwf    FSR
      movlw    "G"
      movwf    INDF
      incf     ptr2,f
```

```
      incf     FSR,f
      movwf    INDF      ;double “G” to indicate start
      incf     ptr2,f    ;next ascii character
      movlw    xfercnt  ;how many bytes to convert to ASCII
      movwf    bitcnt
      movlw    h’4’
      movwf    PCLATH   ;set up PCLATH for lookup table
asciiconv
      movf     ptr1,w
      movwf    FSR
      swapf    INDF,w
      andlw    h’f’       ;isolate the MSN
      call     hex2ascii
      movwf    temp       ;hold the ASCII character
      movf     ptr2,w
      movwf    FSR
      movf     temp,w     ;store ASCII representation of received byte MSN
      movwf    INDF
      incf     ptr2,f     ;advance ASCII ptr
      movf     ptr1,w     ;back to received bytes
      movwf    FSR
      movf     INDF,w
      andlw    h’f’       ;isolate the LSN
      call     hex2ascii
      movwf    temp
      movf     ptr2,w
      movwf    FSR
      movf     temp,w     ;store ASCII representation of received byte LSN
      movwf    INDF
      incf     ptr2,f     ;advance ASCII ptr
      incf     ptr1,f     ;advance received byte ptr
      decfsz   bitcnt,f
      goto     asciiconv
;done data conversion, now indicate newline before sending
      movlw    “\n”       ;newline character
      incf     FSR,f
      movwf    INDF



;cleared for RS232 transmission
      goto     alphabet

;hexadecimal to ASCII conversion table
      org      h’3ff’
hex2ascii
      addwf    PCL,f
      retlw    “0”   ;ascii 0
      retlw    “1”   ;ascii 1
      retlw    “2”   ;ascii 2
      retlw    “3”   ;ascii 3
      retlw    “4”   ;ascii 4
      retlw    “5”   ;ascii 5
      retlw    “6”   ;ascii 6
      retlw    “7”   ;ascii 7
      retlw    “8”   ;ascii 8
      retlw    “9”   ;ascii 9
      retlw    “A”   ;ascii A
      retlw    “B”   ;ascii B
      retlw    “C”   ;ascii C
      retlw    “D”   ;ascii D
      retlw    “E”   ;ascii E
      retlw    “F”   ;ascii F

        end
```

NOTES:

Contact Programmer

1.0 INTRODUCTION

1.1 Chapter Overview

This chapter will address the details of programming the MCRF355 device by using the MCRF355 Contact Programmer. All of the timing diagrams for the device test modes and other specifications can be found in the MCRF355/360 data sheet (DS21287) under Section 3.4, “Signal Timing.” A detailed description of the test modes will not be included in this section. Please refer to the data sheet for more information.

Included in the DV103003 Development Kit is the MCRF355 Contact Programmer. This circuit offers one possible solution for programming the MCRF355 using a PIC16C63. The hardware details, firmware listing, and PC interface are provided here.

1.2 Device Programming

The MCRF355 data array is made up of 154 EEPROM bits. These bits are directly accessed through the device test modes. When erased, each EE cell goes to logic ‘1’. A programming pulse width (T_{WC}, as described in the data sheet) is required to bring each individual bit down to a logic ‘0’.

On the command side, three logic levels are required to program the device: V_{DD}, V_{HH}, V_{IH}, V_{IL}.

V_{DD}	2.5V
V_{HH}	20V typical
V_{IH}	.7 * VDDT Min
V_{IL}	.3 * VDDT Max

Section 3.0 in the data sheet is also titled “Device Programming.” The information contained in this section can also be found there. The test mode commands that are required to program the MCRF355 are as follows:

Test Modes

1. Erase EE
2. Program EE
3. Read EE

The MCRF355 Contact Programmer is designed to carry out these test modes based on commands coming through the RS-232 communications port. The three voltage levels (V_{DD}, V_{HH}, V_{IH}, V_{IL}) are present on the programmer board, originating from the 24 VDC power supply. The test mode timings have been programmed into the PIC16C63, leaving a simple RS-232 command set to initiate and complete the required test modes.

2.0 HARDWARE

2.1 Description of Operation

Programming is initiated on the PC side. The MCRF355 Contact Programmer receives a command from the PC through the RS-232 port, and into the USART on the P16C63. Based on this command, the microcontroller then sends the proper test mode signal to the MCRF355 device, using the three voltage levels present on the programming board. Due to the high voltage level of V_{HH}, additional circuitry was required to buffer these signals from the port pins of the PIC16C63. The switching of the three voltage levels is handled by the an analog switch. The complete schematic for the MCRF355 Contact Programmer can be found in the Section 5.0.

3.0 FIRMWARE

3.1 Overview

Control of the MCRF355 Contact Programmer is accomplished through simple ASCII commands. This enables the user to come up with a PC interface specific to his/her needs. The development kit includes RFLAB, a visual basic interface to the MCRF355 Contact Programmer. However, if a user interface is to be designed that uses the MCRF355 Contact Programmer board and existing firmware, operation of the firmware must be understood. It should be noted here that the MCRF355 Contact Programmer can easily be used in a number of different ways, depending on the operation of the firmware. The included schematic and BOM allows for this approach. This section outlines the operation of the firmware that is included with the development kit.

3.1.1 COMMANDS

The simple ASCII commands mentioned above are described here:

TABLE 3-1: RS-232 COMMANDS, 9600 BAUD, N81

ASCII	Description
P	Program device
E	Erase device
R	Read device
D	Receive data from PC
U	Upload data to PC

Any simple terminal program can be used to interface to the MCRF355 Programmer board. It should be understood that the PICmicro® programs the MCRF355 from data stored in RAM. Updating this data from the PC is accomplished through the serial commands. This data in RAM is described as the PICmicro data array. The commands are initiated with the above ASCII letter. A complete description of each command is included here:

- **Program Device**

An uppercase “P” initiates this command. This command programs the MCRF355 with the data that has been previously loaded into the PICmicro’s data array. The busy light on the MCRF355 Contact Programmer board will go on, showing that programming has been initiated. Upon program completion, the light will go off, and an uppercase “Y” or an uppercase “N” will be echoed back to the PC. The letter “Y” corresponds to a successful program, the letter “N” indicates a failed program.

- **Erase Device**

An uppercase “E” initiates this command. The busy light on the MCRF355 Contact Programmer board will go on, showing that the erase command has been initiated. Upon completion of the erase, the light will go off, and an uppercase “Y” or an uppercase “N” will be echoed back to the PC. The letter “Y” corresponds to a successful erase, the letter “N” indicates a failed erase.

- **Read Device**

An uppercase “R” initiates this command. The busy light on the MCRF355 Contact Programmer board will go on, showing that the read command has been initiated. This command updates the data array in the PIC16C65A with data from the MCRF355. Upon completion, the light will go off, and an uppercase “R” will be echoed back to the PC.

- **Receive Data**

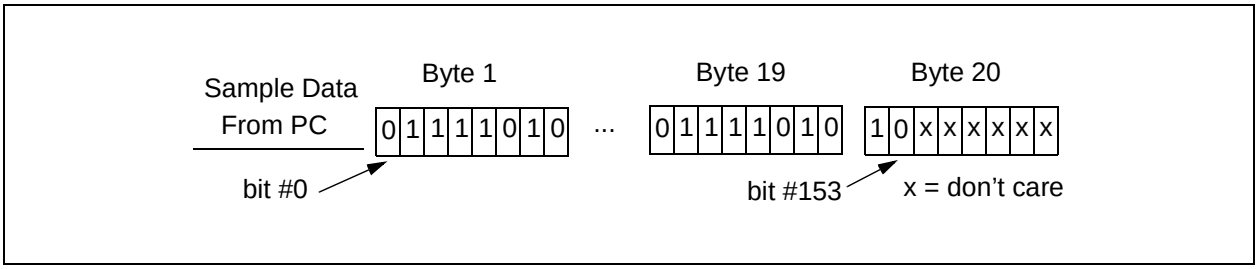
An uppercase “D” initiates this command. This command loads the PIC16C65A data array with the 154 data bits to be programmed. Following the uppercase “D,” the PICmicro will expect 20 more bytes of data. the PICmicro will echo back after each byte. After the 20th byte, the busy light will go off, and the command is complete.

- **Upload Data**

An uppercase “U” initiates this command. This command returns the 154 data bits from the data array in the PICmicro to the PC. Following the uppercase “U,” a single space (ASCII 0x20) will initiate the data transfer. 20 bytes will follow. After the 20th byte, the busy light will go off, and the command is complete.

It should be noted here that the 154-bit data array is not evenly divisible by an 8-bit byte. The 20th data byte contains bits 152 and 153.

FIGURE 3-1: DATA BYTE 20



3.1.2 COMMAND SEQUENCE

Using these five ASCII commands, the user is then able to program and verify the MCRF355 device. Using these five commands, the recommended command sequences follow:

Program

1. **Erase the Device.** Doing an erase command prior to programming is necessary to return all bits to the logic '1' state. Command "E."
2. **Load the Data Array.** The user must send the 20 bytes containing the 154 bits from the PC to the MCRF355 Programmer. Command "D."
3. **Program the Device.** The user must initiate programming. Command "P."

If an uppercase "Y" is returned, the device has been programmed.

4.0 PC INTERFACE

4.1 Overview

Included with the DV103003 developers kit is RFLAB 13.56, a Microsoft® Windows®-based program that handles the above described serial communication. This is a Visual Basic® interface that allows easy control of the MCRF355 Programming board. Figure 4-1 is a screen shot of this software, RFLAB 13.56.

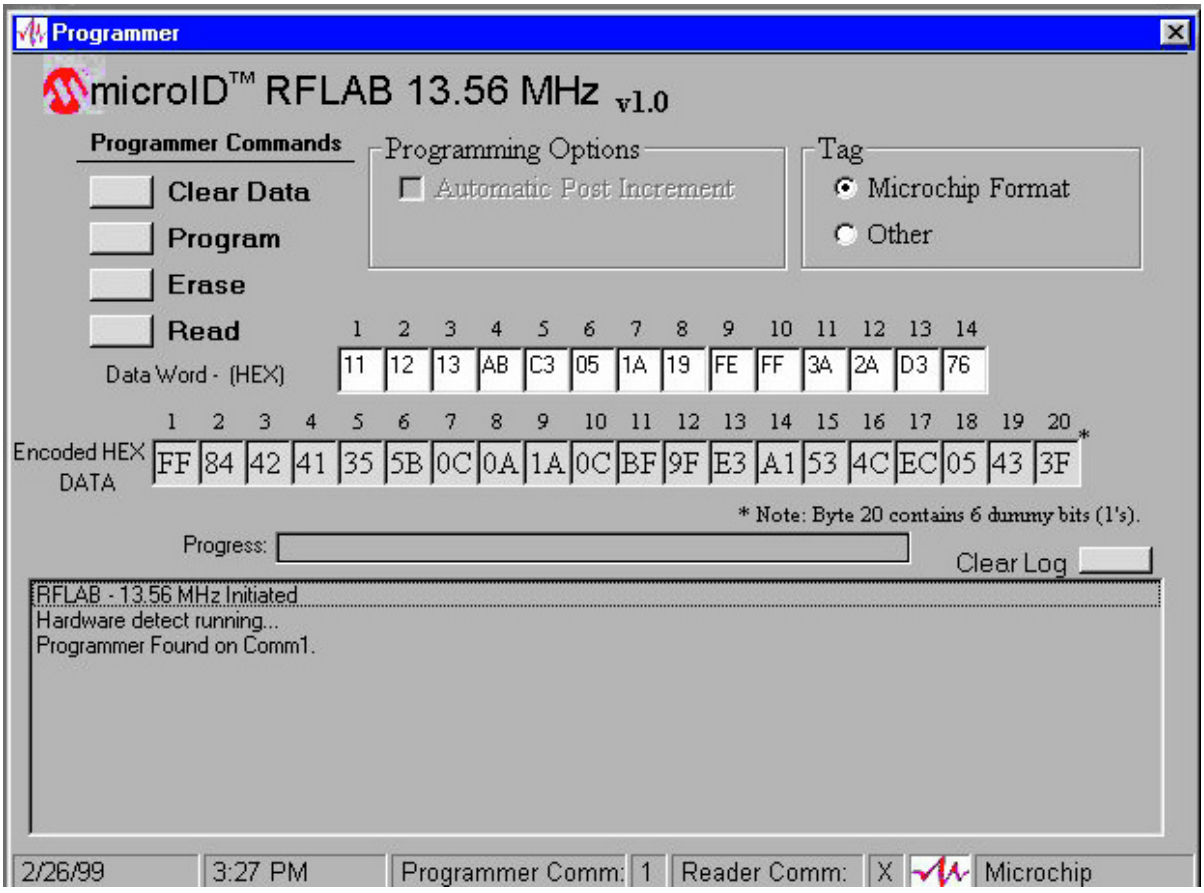
4.2 System Requirements

RFLAB is a 32-bit application developed using Visual Basic 5.0. It will only run under Windows 95 or higher.

4.3 Installation

RFLAB 13.56 MHz comes on two 3.5-inch disks. The entire installation will require approximately 3 megabytes of space. Running a:/setup.exe on disk number 1 will install the software onto your Windows 95 or higher PC. After installation is complete, a shortcut to the executable can be found on your start menu. The default path for this shortcut is under *Program Files>RFLAB> RFLAB 13.56 MHz*.

FIGURE 4-1: RFLAB 13.56 DIALOG



4.4 Operation

Detecting the Programmer

In order for the software to run, the MCRF355 Contact Programmer must be connected to the PC through an available COM port. Please have the programmer connected to the PC, and powered up before you initiate RFLAB. It is important that the 24 VDC power supply be used. When RFLAB comes up, it will scan all of the available serial ports, looking for the MCRF Contact Programmer. Once the programmer is found, the status bar at the bottom of the window will indicate this. If the programmer is not found, try selecting the COM port manually instead of relying on the auto detection process. Manual selection of the COM port can be done by choosing which COM port on the menu: *Options>COM Port*. If this also fails, please verify that all power connections and serial port connections are secure. Once the software has detected that a programmer is connected to the computer, you are free to program a device.

4.4.1 COMMAND BUTTONS

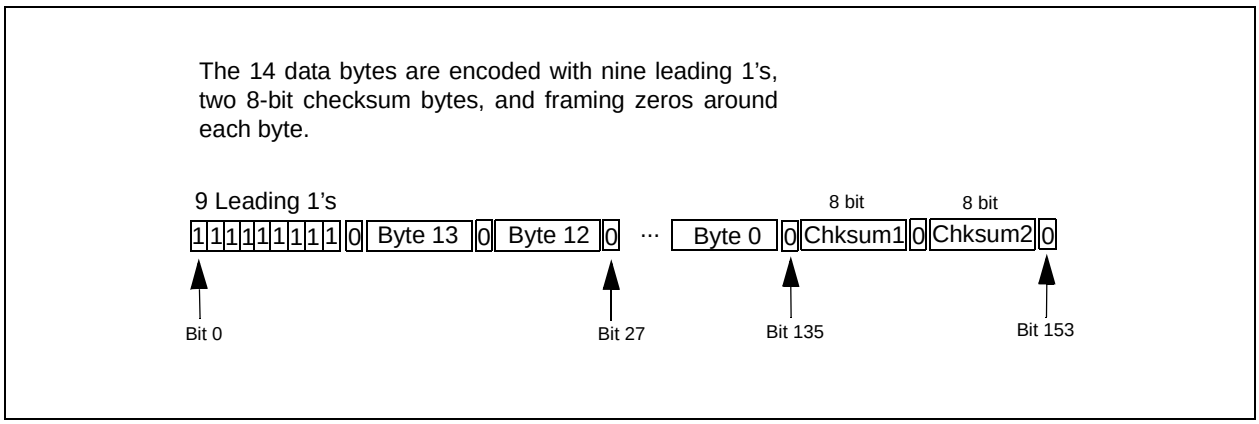
The top left corner of the window will show you four command buttons. Clear Data, Program, Erase, and Read. The CLEAR DATA button will clear the text boxes in the window. This command will not erase the device, it just clears the text boxes on your screen. The ERASE button will send the erase command to the programming board. The PROGRAM button will initiate the programming sequence described above in Section 3.1.2. A single click will Erase the device, download the data, program the device, and then verify the program. The READ button will send a Read command, followed by an Upload command to the programming board. The 20 text boxes on the screen will then be updated with the correct data.

The bottom third of the programming window shows a list box. This list box will echo the commands as they are being sent to the MCRF355 Contact Programming board. All of the command buttons described will log activity in this box. This box is provided to better describe the ASCII commands behind the command buttons.

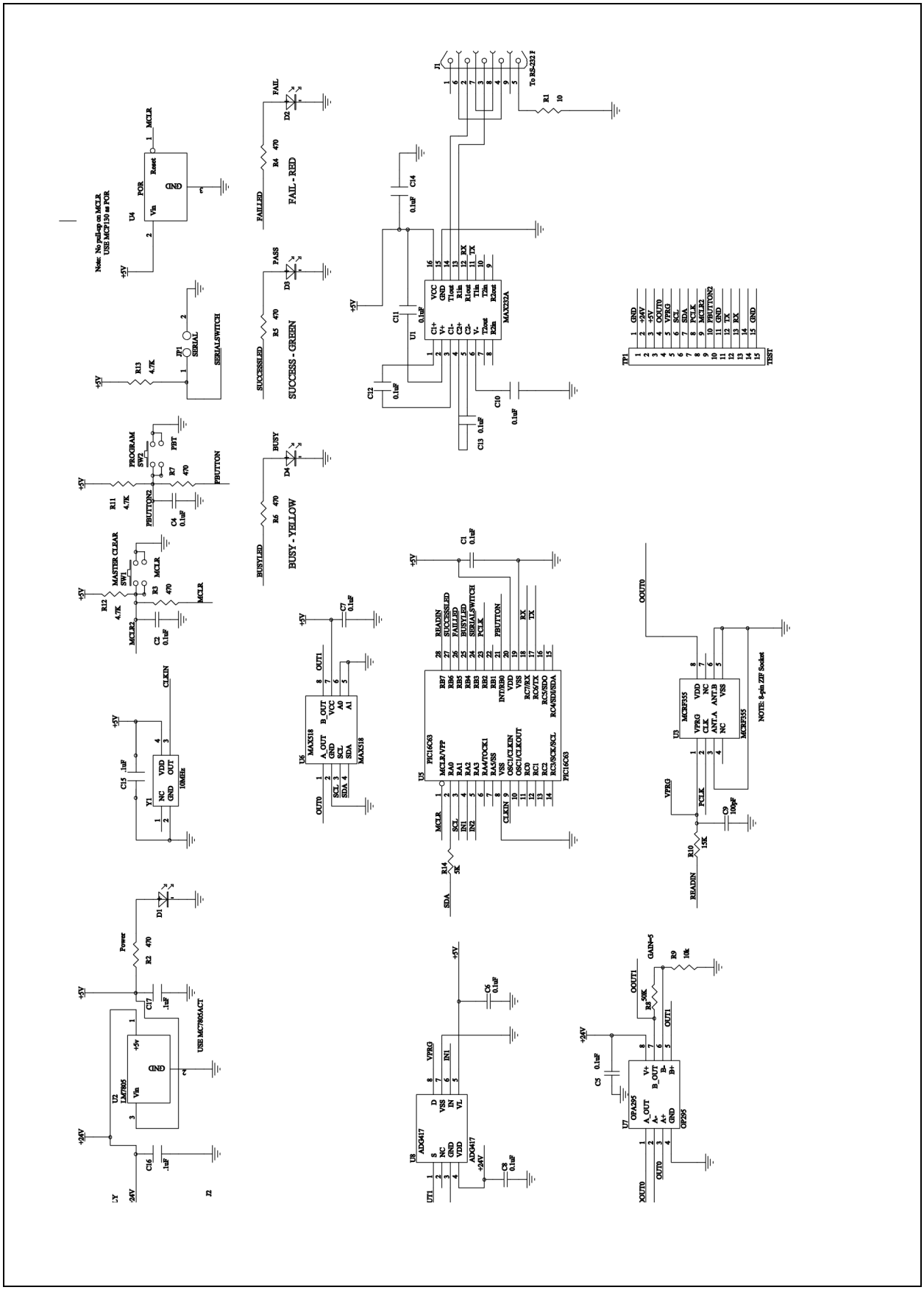
4.4.2 DATA FORMAT

The top right corner of the programming window will show you a selector labeled TAG. The two selections are marked “Microchip” and “Other.” This selector toggles the way the data is encoded into the 154-bit stream. When the selector is set to “Other,” the user has complete control over each bit. The 20 test boxes labeled HEX DATA are free to be edited. When the selector is set to “Microchip,” only 14 of the 20 data bytes are free for edit. An additional 14 text boxes will pop up showing this. This mode encodes these 14 data bytes into a 154 bit stream. Nine header bits, two checksum bytes, and framing zeros are encoded along with these 14 data bits. This encoding scheme is detailed in Figure 4-2. The “Microchip” format is used by firmware in the 13.56 MHz reference reader, included in the DV103003 kit.

FIGURE 4-2: MICROCHIP TAG FORMAT



5.0 CONTACT PROGRAMMER SCHEMATIC



6.0 CONTACT PROGRAMMER BILL OF MATERIALS

Assembly #	Qty	Part #	Manufacturer	Part Description	Reference Designator
02-01524	1	02-01524-D		PCB ASSY DWG, MCRF355 microID Programmer	
02-01524	1	03-01524		SCHEMATIC, MCRF355 microID Programmer	
02-01524	1	04-01524		PCB FABRICATION, MCRF355 microID Programmer	
02-01524	1	PIC16C63A-04/P	MICROCHIP	IC, PIC16C63A-04/SP, 8 BIT CMOS MICROCONTROLLER 28P	U5
02-01524	1	110-91-328-41-001	MILL-MAX	SOCKET, COLLET OPEN FRAME 28P .300W	xU5
02-01524	1	MC7805ACT	MOTOROLA	IC, +5V REG 0.5A	U2
02-01524	1	AC-001	POWER DYNAMICS	JACK, POWER, 3 PIN, PC MOUNT	J2
02-01524	1	521-9173	DIALIGHT	LED, GREEN T1-3/4 DIFFUSED	D3
02-01524	2	521-9174	DIALIGHT	LED, YELLOW T1-3/4 DIFFUSED	D1, D4
02-01524	1	OECS-100-1-A101A	ECS	XTAL OSC, 10.000 MHZ, 14P FULL SIZE	Y1
02-01524	1	MAX232ACPE	MAXIM	IC, HS 5V DUAL RS232 DRIVER 16 DIP	U1
02-01524	1	MCP130-450DI/TO	MICROCHIP	IC, TO 92, SUPERVISOR CIRCUIT W/OPEN DRAIN OUTPUT	U4
02-01524	1	KF22-E9S-NJ	KYCON	CONN, D-SUB 9P RECPT RT ANGLE WITH JACK SCREWS	J1
02-01524	1	2-0016-03340-000-006-00X	3M TEXTOOL	SOCKET, TEST, 16P DIP (0.300) GREEN	xU3
02-01524	15	592CZ5U104M050B	SPRAGUE	CAP, CER AXIAL 0.1uF 50V	C1,C2,C4-C8,C10-C17
02-01524	6	470QBK	YAGEO	RES, 470 OHM 1/4W 5% CARBON FILM RES	R2-R7
02-01524	1	10QBK	YAGEO	RES, 10 OHM 1/4W 5% CARBON FILM RES	R1
02-01524	2	B3F-1000	OMRON	KEYSWITCH, MOMENTARY PCB MOUNT	SW1, SW2
02-01524	4	SJ-5018	3M	MISC, RUBBER FEET, .50 SQ .23 HIGH BLACK	Placed at corners of PCB
02-01524	1	08-00126		LABEL, NEED HELP WITH ASSY/SERIAL	
02-01524	1	08-00171		LABEL, MCRF355 PROGRAMMER FIRMWARE, 355_9.HEX, x/xx/99, U1, CS: xxxxh	@ U1
02-01524	1	TSW-1-15-07-G-S	SAMTEC	HEADER, 1x15, BREAKAWAY, 0.025 SQ POST, GD CONTACT	TP1
02-01524	1	577202B00000	AAVID	HEATSINK, TO-220, 0.5	@U2
02-01524	1	P13		SCREW, #4-40 x 3/8 PANHEAD PHILLIPS	@U2

Assembly #	Qty	Part #	Manufacturer	Part Description	Reference Designator
02-01524	1	701-7357	CONCORD	NUT, #4-40 HEX STEEL CAD-PLATED	@U2
02-01524	1	ADG417BN	ANALOG DEVICES	IC, ADG417BN ANALOG SWITCH, 8P DIP	U8
02-01524	1	MAX518	MAXIM	IC, MAX518 8-BIT DAC, 8P DIP	U6
02-01524	1	OP295	ANALOG DEVICES	IC, OP295 DUAL OP AMP	U7
02-01524	1	521-9165	DIALIGHT	LED, RED T1-3/4 DIFFUSED	D2
02-01524	1	TSW-102-07-S-S	SAMTEC	HEADER, 2 PIN 0.025 SQ POST	JP1
02-01524	1	MNT-102-BK-T	SAMTEC	2 POSITION JUMPER BLOCK (SHUNT)	@JP1
02-01524	1	A470J15C0GHVVWA	PHILLIPS	CAP, 47pF AXIAL CERAMIC C0G 100V 5%	C9
02-01524	2	5043CX5K100J	PHILLIPS	RES, CF 5.1K 5% 1/4W	R10,R14
02-01524	1	MFR-25FBF 22K1	YAGEO	RES, 22.1K OHM 1/4W 1% METAL FILM RES	R8
02-01524	1	MFR-25FBF 2K21	YAGEO	RES, 2.21K OHM 1/4W 1% METAL FILM RES	R9

7.0 CONTACT PROGRAMMER SOURCE CODE FOR THE PICmicro® MCU

```
#include <P16c63.inc>    ; include file for the processor

         TITLE “MCRF355 Programmer Board”
         __config b’00000001000001’  ;protection off,PWRT disabled,watchdog disabled,XT oscilla-
tor

; Note: Assume 10 Mhz Crystal -=> 1 instruction = 400ns, .4us

         list p=16c63

;The purpose of this firmware program is to accept commands via RS-232 and execute
;a set of commands on the MCRF355 (DUT).  The device is serially programmed using
;the MCRF clock dutsck and data dutsda.  The dut Vcc is generated by an 8-bit DAC
;through a unity gain buffering Op-Amp.  The DUT programming voltage is also generated
;by an 8-bit DAC through a buffering Op-Amp with a gain of 11.  An analog switch
;is used to isolate the Op-Amp Vpp from the dutsda line, as they are multiplexed.
;The program is structured to conduct read of FF after erasing.  The program also
;goes directly to the read mode after programming.  MCRF355 device serial test modes
;are used for erase, read, and program as follows:

;TEST MODE CODES FOR THE MCRF355
#define erase_code b’11010100’
#define read_code  b’11010110’
#define prog_code  b’11010010’


;Serial Commands from PC
;-----------------------

;    E    -  Send ERASE command to TAG
;    U    -  Upload data array to PC
;    D    -  Download data array from PC
;    P    -  Send PROGRAM command to TAG
;    R    -  Send READ command to TAG
;   *** any unknown command will result in an error condition, in which the PICmicro will
;    return a lowercase ‘e’ to the PC and flash the busy light for 2 seconds.

;General program description:

; Five of the above commands/routines (Erase, Read, Program, Upload, and Download) can be used
; to program/erase/read the TAG. The Program / Read commands update the data_arry in the picMicro.
; The following are examples of how one would erase/read/write to the TAG:


;Erase Sequence:
; 1. “E” - Send the erase command to the PART
; 2. “R” - update the picMicro data_array


;Read Sequence:
; 1. “R” - update the picMICRO data_aray
; 2. “U” - upload this data array to the PC


;Program Sequence: (includes an erase, erase verify, and program verify)
; 1. “E” - Send the erase command to the PART
; 2. “R” - update the picMicro data_array
; 3. “D” - download data to be programmed
; 4. “P” - Program the TAG
; 5. “R” - Read back from the TAG
; 6. “U” - upload and verify all bits programmed.
```

```
;* for multiple programs of a known blank part, the command/routine “P” is all that would be
required.

;Definitions are as follows:

#define dutpgm                  portb,0     ; push-button programming
#define dutsck                  portb,2     ; serial clock for dut
#define serial_switchportb,3                ; junper for serial programming
#define led_yellowportb,4                   ; BUSY indicator
#define led_red                 portb,5     ; Fail indicator
#define  led_green              portb,6     ; Pass indicator
#define dutsda                  portb,7     ; serial data for clock

#define sda                     porta,0     ; DAC serial data
#define scl                     porta,1     ; DAC serial clock
#define  dutvpp                             porta,2; Analog Switch b enable for Vpp

#define which_dacflag,7                     ;This flag is used to select either
                                            ;DAC 0 or 1
#define current_bitflag,0

;These are 8-bit DAC, vref = 5.0V voltages going through op-amp stages
;vhh has gain of 11X, vcc has gain of 1X

#define vil.0           ;will get you 0 volts


;Memory Allocation
;-----------------
                        cblock     0x20


rcv_data  ;UART data
read_byte ;Storage location during read_device
temp                    ;TEMP variable
temp1                   ;Temp variable
temp2                   ;TEMP variable
tempa                   ;TEMP variable
tempb                   ;TEMP variable
tempd1                  ;TEMP variable
tempd2                  ;TEMP variable

mcrf_bottom :20;bottom of the entire data spot
pc_bottom:20;bottom of the entire data spot, used for verify.
temp3                   ;TEMP variable
ucount                  ;TEMP variable
flag                    ;TEMP variable

vhh                     ;Vpp Programming voltage for DUT
vcc                     ;Vcc voltage for DUT

daber1                  ;TEMP variable
daber2                  ;TEMP variable
Vdut                    ;dut Vpp  variable
                        endc

;        flag
;------------------------------
;7 - which DAC is being SET
;6
;5
;4
;3
;2
;1
;0 - used for BIT toggling
```

```
;



;MAIN ROUTINE

                        org         0x00    ;RESET VECTOR
RR                      goto        reset

                        org         0x05    ;Program Memory Begin



;Set up the Data Direction Registers Port B
reset    bsf     STATUS,RP0      ;bank1
         clrf                PORTB               ;Clear pins
         movlw   B'00000001'     ;
         movwf   TRISB           ;set DDR- portB
         bcf                 status,rp0          ;
         clrf                PORTB               ;Clear pins

;Set up the Data Direction Registers Port A
         bsf     STATUS,RP0      ;bank1
         movlw   B'00000000'     ;
         movwf   TRISA           ;set DDR- portB
         bcf                 status,rp0          ;
         clrf                PORTA               ;Clear pins

;Set up the UART

         bsf                 STATUS,RP0          ;Select register page 1
         movlw               b'00100000'         ;Enable RCIF interrupt
         movwf               PIE1
         movlw               .64                 ;9600 baud @10MHz
         movwf               SPBRG
         movlw               b'10100100'         ;Async, High baud rate
        movwfTXSTA
         bcf                 STATUS,RP0          ;Select register page 0
         movlw               b'10010000'         ;Enable continous reception
         movwf               RCSTA             ; 8-bit,spen=1,cren=1 to enable rcv
         movlw               b'01000000'         ;disable global interrupts
         movwf               INTCON

;Reset the DACs to 0V and open the analog switch

         bsf             dutvpp      ;open analog switch
         bsf             flag,7      ;DAC 1 to 0V
         movlw           vil
         movwf           vdut
         call            SetDACXV

         bcf             flag,7      ;DAC 0 to 0V
         movlw           vil
         movwf           vdut
         call            SetDACXV

;Set up variables

         movlw           .98
         movwf           vhh         ;Set Vhh to .98 in 8-bit DAC with gain of 11 ~ 20V
         movlw           .128
         movwf           vcc         ;Set DUT vcc to 2.5V, 8-bit DAC with unity gain

main
```

```
; This is the main loop. The PICmicro will sit here and
; wait for a byte to come in from the PC.


wait
         bcf            led_yellow ;Turn off busy light
waitloop
        btfss   pir1,rcif;Check if a byte has come into UART
         goto    waitloop ;has the pattern arrived yet?

         movf           rcreg,0           ;move rcreg into w
         movwf          rcv_data   ;move w into rcv_data
         bcf            led_green
         bcf            led_red
         bsf            led_yellow


;Execute when data received in UART

cchk1
         movlw   0x50            ; 'P'
         xorwf   rcv_data,0      ; exclusive OR against the command
         btfss          status,2        ; if ZERO bit is set, they were the same.
         goto           cchk2              ; False
         call           program_device; True
         movlw          0x59               ; "Y" for true
         btfss          flag,1             ; test for programming success
         movlw          0X4E               ; "N" for false
         movwf   txreg           ; ack command complete to pc
         goto    wait


cchk2   movlw   0x45            ; 'E'
         xorwf   rcv_data,0      ; exclusive OR against the command
         btfss          status,2        ; if ZERO bit is set, they were the same.
         goto           chk3               ; False
         call           erase_device; True
         movlw          0x59               ; "Y" for True
         btfss          flag,1             ;  Test for Erase Success
         movlw          0x4E               ; "N" for False
         movwf   txreg           ; ack command complete to pc
         goto    wait


chk3    movlw   0x52            ; 'R'
         xorwf   rcv_data,0      ; exclusive OR against the command
         btfss          status,2        ; if ZERO bit is set, they were the same.
         goto           chk4               ; False
         call           read_device; True
         movlw          0x52
         movwf   txreg           ; ack command complete to pc
         goto    wait


chk4    movlw   0x55            ; 'U'
         xorwf   rcv_data,0      ; exclusive OR against the command
         btfss          status,2        ; if ZERO bit is set, they were the same.
         goto           chk5               ; False
         call           upload             ; True
         goto    wait


chk5    movlw   0x44            ; 'D'
         xorwf   rcv_data,0      ; exclusive OR against the command
         btfss          status,2        ; if ZERO bit is set, they were the same.
         goto           chk6               ; False
```

```
         movlw          0x44
         movwf   txreg           ; ack command complete to pc
         call           download   ; True
         goto    wait


chk6     movlw          0x20              ; " " space
         xorwf          rcv_data,0
         btfss          status,2
         goto           errr              ; False
         movlw          0x53
         movwf   txreg           ; ack command complete to pc
         goto           wait


;**************
;*program EEPROM*
;**************
; Sends the program command sequence to the TAG
; and then program from the PICmicro's data array (mcrf_bottom).
; 5/11/99
;
program_device

         Call           dutStart          ;Give dut start bit

         movlw          prog_code         ;load the proper opcode
         movwf          temp              ;set the opcode to be sent
         Call           send_opcode       ;send the opcode


         movlw          pc_bottom
         movwf          FSR
         movlw          .20               ;20 bytes/word
         movwf          temp

llpm2    movf           INDF,0            ;BYTE LOOP
         movwf          temp2

         movlw          .8
         movwf          tempb             ;8 bits/byte

anotherbit
         btfss          temp2,7           ;BIT LOOP
         Call           pgm_pulse

         rlf            temp2,1
         bsf            dutsck            ;send a clock to get it on the next bit
         call           dly_4uS
         bcf            dutsck
         call           dly_4uS

         decfsz         tempb,f           ;all BITS finished?
         goto           anotherbit        ;No, goto bit look

         incf           FSR,f
         decfsz         temp,1            ;all BYTES finished?
         goto           llpm2             ;No, goto byte loop


         Call           dutSTOP           ;Give dut stop bit



;**************
;*Read EEPROM*
```

```
;**************
; Sends the read command sequence to the TAG
; and then puts the data into the PICmicro's data array (mcrf_bottom).
; 5/11/99
;
read_device

         Call           dutStart                    ;Give dut start bit

         movlw          read_code                   ;load the proper opcode
         movwf          temp                        ;set the opcode to be sent
         Call           send_opcode                 ;send the opcode


;now READ in the data and put it into RAM (starting at location mcrf_bottom)

         bsf     STATUS,RP0      ;bank1
         movlw          b'10000001'    ;!
         movwf          trisb                       ;switch port to an INPUT
         bcf     STATUS,RP0      ;bank0

         movlw          mcrf_bottom    ;
         movwf          FSR                         ;set pointer to bottom of data Array
         movlw          .20                         ;
         movwf          temp                        ;number of bytes to the end

read_byteloop                           ;BYTE LOOP

         movlw          .8
         movwf          tempb
         movlw          0xFF
         movwf          temp2                       ;default data to all 1's

read_bitloop                            ;BIT LOOP
         rlf            read_byte,1                 ;First rotate is dummy
         call           read_one_bit   ;Go and READ the VPRG LINE
         call           clock_mcrf     ;
         decfsz         tempb,1                     ;FINISHED WITH BITS?
         goto           read_bitloop   ;ROLL THE BITS, GOTO BIT LOOP

         movf           read_byte,0                 ;data to be put into pic memory
         movwf          INDF
         incf           FSR,f
         decfsz         temp,1                      ;FINISHED WITH BYTES?
         goto           read_byteloop  ;NO, GOTO BYTE LOOP

         Call           dutSTOP                     ;Give dut stop bit


;******************
;Verify
;******************
         bsf            flag,1                      ;reset flag
         movlw          .19                         ; 20 byte word
         movwf          ucount                      ;

         movlw          MCRF_bottom
         movwf          temp                        ; first counter
         movlw          PC_bottom
         movwf          temp2                       ; second counter

topoloop

         movf           temp,0                      ; get first place
         movwf          FSR
         movf           INDF,0                      ; get first data
```

```
         movwf          daber1             ; store first data
         movf           temp2,0            ; get second place
         movwf          FSR
         movf           INDF,0             ; get second data
         movwf          daber2             ; store second data

;now compare the two data sets: daber1, and daber2

         incf           temp,1             ;increase the two pointers
         incf           temp2,1            ;increase the two pointers

         movf           daber1,0
         XORWF          daber2,0
         btfss          status,2
         goto           false
         decfsz         ucount,1
         goto           topoloop
         goto           BITSTWO

false
         bcf            flag,1
         bsf            led_red            ;turn on fail led
         decfsz         ucount,1
         goto           topoloop


;now compare the final two bits (byte 20)
;temp and temp2 are ALREADY on the 20th byte
bitstwo
         movf           temp,0
         movwf          FSR
         movf           INDF,0
         movwf          daber1
         movf           temp2,0
         movwf          FSR
         movf           INDF,0
         movwf          daber2
         movlw          b'11000000' ;mask off the bottom 6
         andwf          daber1,1
         andwf          daber2,1
         movf           daber1,0   ;now compare
         XORWF          daber2,0
         btfss          status,2
         goto           false2
         btfsc          flag,1             ; test for programming success
         bsf            led_green  ;turn on pass led
         return

false2
         bcf            flag,1
         bsf            led_red            ;turn on fail led
         return

         RETURN




; Read and Flag bit either 1 or 0

read_one_bit
         bsf            read_byte,0
         btfss          dutsda
         bcf            read_byte,0
         return
```

```
;**************
;*Erase EEPROM*
;**************
; Sends the erase command sequence to the TAG
; 5/11/99
;
erase_device

         Call           dutStart           ;Give dut start bit

         movlw          erase_code         ;load the proper opcode
         movwf          temp               ;set the opcode to be sent
         Call           send_opcode        ;send the opcode

         Call           pgm_pulse

         Call           dutSTOP            ;Give dut stop bit


;adding the erase verify
;fill the obottom with 0xff

         movlw          .20
         movwf          ucount
         movlw          PC_bottom
         movwf          FSR
elp1
         movlw          0xFF
         movwf          INDF
         incf           FSR,1
         decfsz         ucount,1
         goto           elp1

         goto           read_device


;***************
;program pulse routine
;***************

pgm_pulse

         bcf            dutvpp             ;Close Analog switch to dut Vpp

         bsf            flag,7             ;Set Dac 1 to be set
         movf           vhh,0              ;Set Vpp = 20V (gain of 10)
         movwf          vdut               ;Load the register
         Call           SetDACXV           ;Turn on the DAC

         Call           d10_10ms

         bsf            flag,7             ;Set Dac 1 to be set
         movlw          vil                ;Set Vpp = 0V
         movwf          vdut               ;Load the register
         Call           SetDACXV           ;Turn on the DAC

         Call           d10_1ms
         bsf            dutvpp             ;Close Analog switch to dut Vpp

         return

;*********************************************
;sends a clock pulse with the vprg line at 0 volts
; Vil pulse (just clock)
;*********************************************
```

```
clock_mcrf
         bsf            dutsck             ;clock pulse
         call           dly_4uS            ;send a clock to ADVANCE to the next bit
         bcf            dutsck
         return



dutStart
;rising edge of clock during a Vhh pulse

         bsf     STATUS,RP0      ;bank1
         movlw   B'10000001'     ;TRI-state the READIN pin of port B
         movwf   TRISB             ;while the START BIT is being
         bcf             status,rp0         ;sent
         ;clrf           PORTB              ;

         bcf            flag,7             ;Set DAC 0 to be set (Dut Vcc)
         movf           vcc,0              ;Set Vcc = 2.5V
         movwf          vdut               ;Move Vcc = 2.5 into variable
         Call           SetDACXV           ;SET the voltage on VCC

         bcf            dutvpp             ;Close Analog switch to dut Vpp
         bsf            flag,7             ;Set Dac 1 to be set
         movf           vhh,0              ;Set Vpp = 20V (gain of 10)
         movwf          vdut               ;Load the register
         Call           SetDACXV           ;Turn on the DAC

         call           d10_1ms
         call           d10_1ms

         bsf            dutsck

         bsf            flag,7
         movlw          vil
         movwf          vdut
         call           SetDACXV

         call           d10_1ms
         call           d10_1ms
         bcf            dutsck

         bsf            dutvpp             ;Open Analog switch to dut Vpp

         call           d10_1ms            ;

         bcf            dutsda             ;make sure dut data line is low
         bsf     STATUS,RP0      ;bank1
         movlw          b'00000001'        ;!
         movwf          trisb              ;switch port to an OUTPUT
         bcf     STATUS,RP0      ;bank0

         return


dutStop
;Vhh pulse during clock high


         bsf     STATUS,RP0      ;bank1
         movlw   B'10000001'     ;TRI-state the READIN pin of port B
         movwf   TRISB             ;while the START BIT is being
         bcf             status,rp0         ;sent
         ;clrf           PORTB              ;

         bcf            dutvpp             ;Close Analog switch to dut Vpp
```

```
        bsf             dutsck
        bsf             flag,7              ;Set Dac 1 to be set
        movf            vhh,0               ;Set Vpp = vhh
        movwf           vdut                ;Load the register
        call            setDACXV            ;Turn on the DAC

        call            d10_1ms
        call            d10_1ms

        bsf             flag,7              ;Set Dac 1 to be set
        movlw           vil                 ;Set Vpp = vil
        movwf           vdut                ;Load the register
        call            setDACXV            ;Turn on the DAC

        call            d10_1ms
        call            d10_1ms
        nop
        bcf             dutsck
        bsf             dutvpp              ;Open Analog switch to dut Vpp

        bcf             dutsda              ;make sure dut data line is low
        bsf     STATUS,RP0      ;bank1
        movlw           b'00000001'         ;!
        movwf           trisb               ;switch port to an OUTPUT
        bcf     STATUS,RP0      ;bank0

        return



;******************
;Send OPCODE
;******************
;This will send the appropriate OPCODE to the MCRF device

send_opcode

        movlw           .8
        movwf           tempb
        CALL            dly_4uS
        BSF             dutsck
        CALL            dly_4uS
        bcf             dutsck
        CALL            dly_4uS
        bsf             dutsda
        CALL            dly_4uS
        bsf             dutsck
        CALL            dly_4uS
        bcf             dutsck
        CALL            dly_4uS
        bcf             dutsda
        CALL            dly_4uS


b2mloop
        bsf             flag,0
        btfss           temp,7
        bcf             flag,0
        Call            Sendbit2mcrf
        rlf             temp,1
        Decfsz          tempb,1
        goto            b2mloop
        Return

Sendbit2mcrf
        btfsc           flag,0
```

```
         bsf            dutsda
         CALL           dly_4uS
         bsf            dutsck
         CALL           dly_4uS
         bcf            dutsck
         CALL           dly_4uS
         bcf            dutsda
         CALL           dly_4uS
         retlw          0


;**************
;*UPLOAD to PC*
;**************
; Sends the data_array to the PC.


upload
                        movlw      .20     ; 20 byte word
                        movwf      ucount  ;
                        movlw      mcrf_bottom;
                        movwf      FSR     ; set pointer to bottom of array
                        call       d10_1ms ;
uwait    btfss   pir1,rcif;
                        goto    uwait       ; waiting for inital handshake (Space)
                        call       d10_1ms ;
                        movf       INDF,0  ;
                        movwf      txreg   ; send the data
                        incf       FSR,f   ; increase pointer
                        call       d10_1ms ;
                        decfsz     ucount,1; all bytes sent?
                        goto       uwait   ;
                        return


;******************
;*Download from PC*
;******************
; Fills the PICmicro data array with data from the PC

download
                        movlw      .20     ; 20 BYTE WORD
                        movwf      ucount
                        movlw      pc_bottom
                        movwf      FSR     ; SET pointer to bottom of array
waitd            btfss   pir1,rcif
                        goto    waitd       ; has the pattern arrived yet?
                        movf    RCREG,0         ; move rcreg into w
                        movwf      INDF
                        movwf      txreg   ; ack it back
                        incf       FSR,f   ; increase pointer
                        decfsz     ucount,1; ALL BYTES received?
                        goto       waitd   ; NO, look again

                        return



errr
;*****************
;*Error condition*
;*****************
; Sends the 'e' back to the PC. This is
; also how RFLAB detects if there is a programmer
; connected on one of the COMM ports.
```

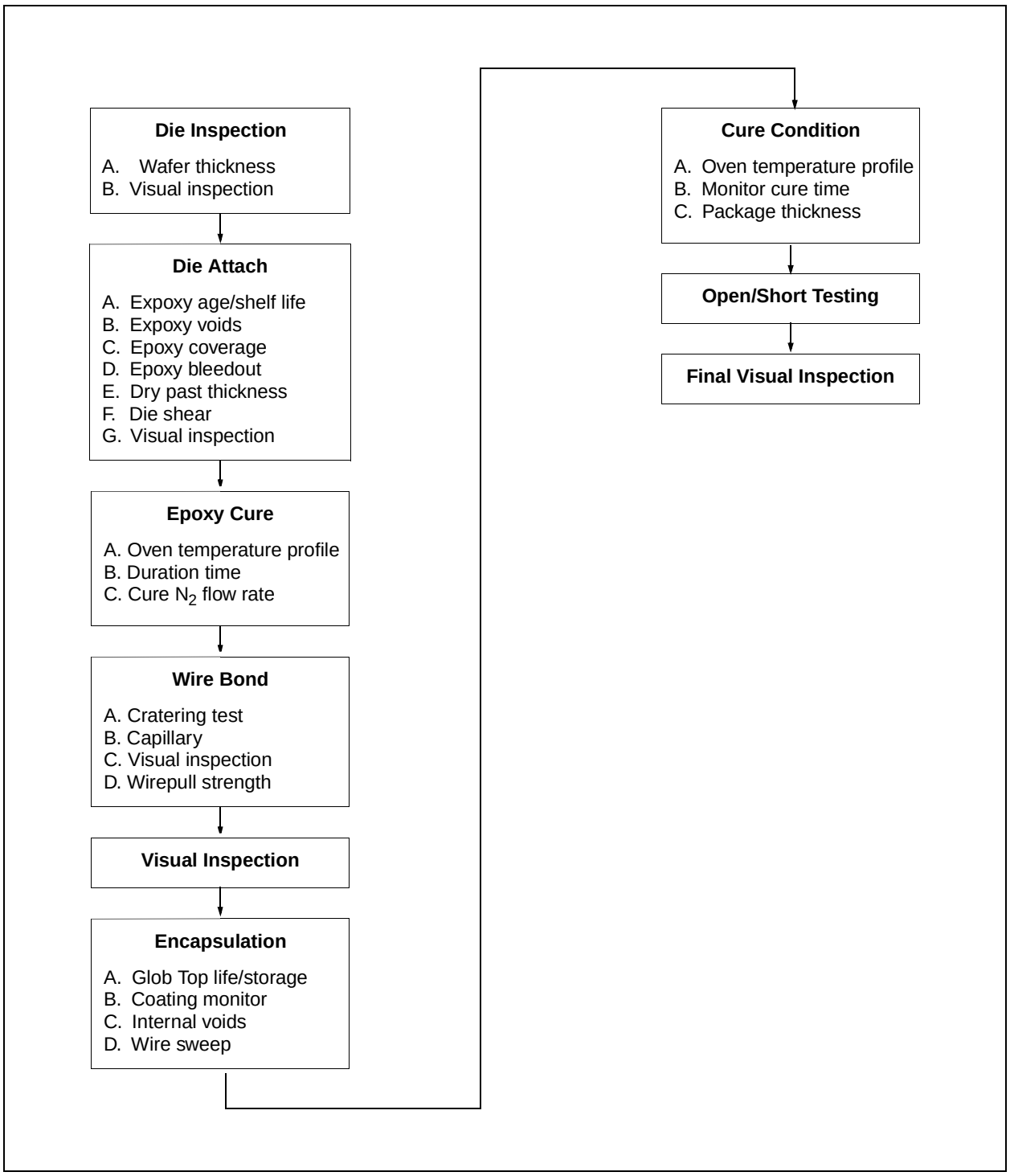
```
;blink the light- something went wrong
         movlw          0x65              ;'e'
         movwf          txreg             ; send it
         call           d10_10ms

         movlw          .18
         movwf          tempa

flm      bsf            led_yellow        ;flash the LED for ~2 seconds (visible error notifica-
tion to the user)
         bsf            led_red
         bsf            led_green
         call           d10_10ms
         call           d10_10ms
         call           d10_10ms
         call           d10_10ms
         call           d10_10ms
         call           d10_10ms
         call           d10_10ms
         call           d10_10ms
         call           d10_10ms
         call           d10_10ms
         bcf            led_yellow
         bcf            led_red
         bcf            led_green
         call           d10_10ms
         call           d10_10ms
         call           d10_10ms
         call           d10_10ms
         call           d10_10ms
         call           d10_10ms
         call           d10_10ms
         call           d10_10ms
         call           d10_10ms
         decfsz         tempa,1
         goto           flm
         goto           wait              ; back to waiting for another command


         return
         goto           wait



SetDACXV
         Call           DacStart
         movlw          b'01011000'
         movwf          temp1
         call           byte2DAC

         Call           Ack

         movlw          b'00000000'
         btfsc          flag,7
         movlw          b'00000001'
         movwf          temp1
         Call           byte2DAC

         Call           Ack

         movf           Vdut,0            ; Change Vdut, DAC0 = dut Vcc, DAC1 = dut Vpp
         movwf          temp1
         Call           byte2DAC

         Call           Ack
         Call           DacStop
```

```
         Return

byte2DAC

         movlw          .8
         movwf          temp3
b2dloop
         bsf            flag,0
         btfss          temp1,7
         bcf            flag,0
         Call           Sendbit2Dac
         rlf            temp1,1
         Decfsz         temp3,1
         goto           b2dloop
         Return



Sendbit2Dac
         bcf            sda
         btfsc          flag,0
         bsf            sda
         bsf            scl
         CALL           dly_4uS
         bcf            scl
         bcf            sda
         CALL           dly_4uS
         retlw          0

Ack      bsf            scl
         CALL           dly_4uS
         bcf            scl
         CALL           dly_4uS
         retlw          0

DacStart

         bsf            sda
         nop
         bsf            scl
         CALL           dly_4uS
         bcf            sda
         CALL           dly_4uS
         bcf            scl
         retlw          0


DacStop

         bsf            scl
         CALL           dly_4uS
         bsf            sda
         CALL           dly_4uS
         bcf            scl
         nop                        ;Delay 400nS each
         bcf            sda
         retlw          0



;******************
;4.4uS eelay routine
;******************

dly_4uS
         movlw          .2                ;
```

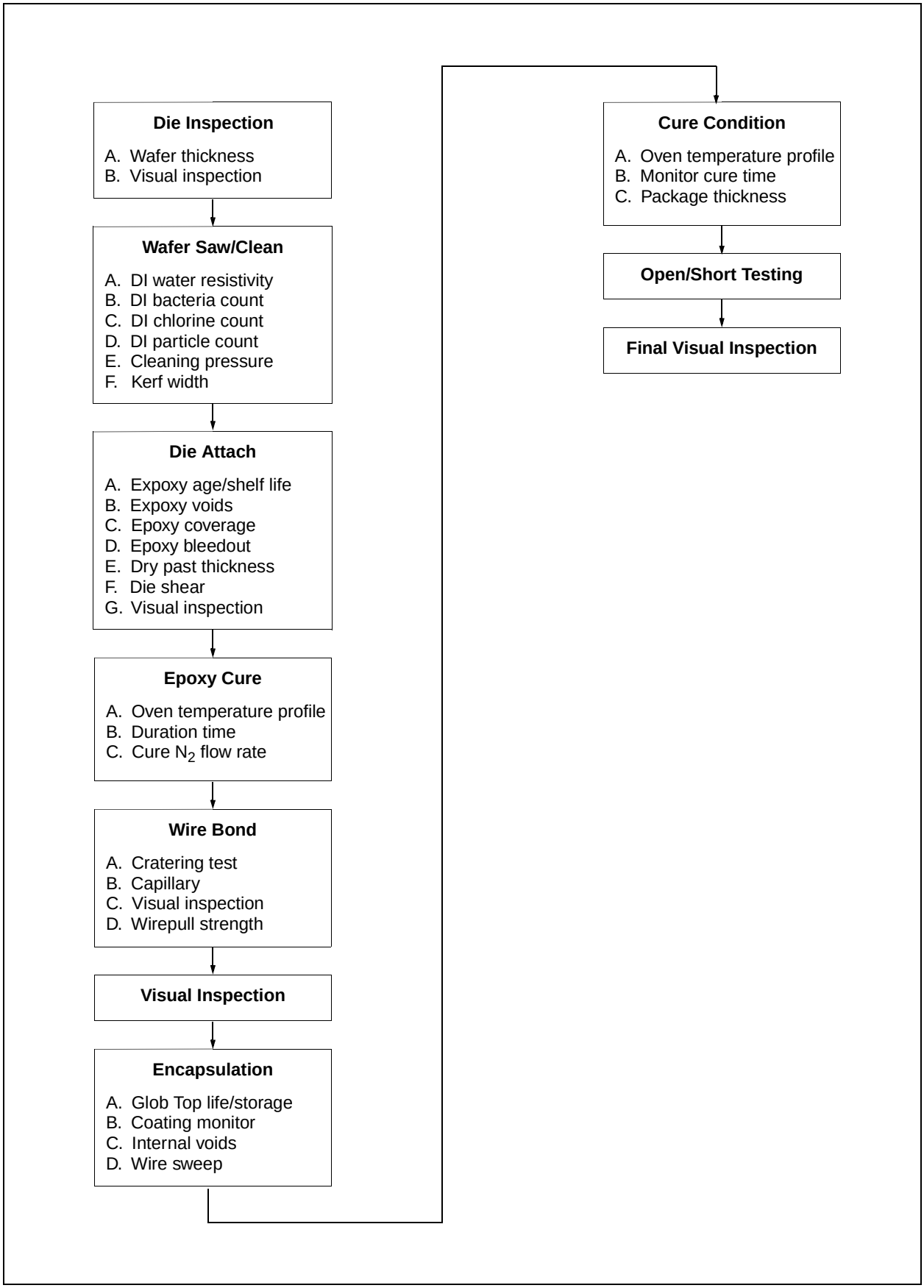
```
         movwf          tempd1            ;
dlyloop4uS
         nop                              ;
         decfsz         tempd1,1   ;
         goto           dlyloop4uS ;
         return                           ;



;******************
;1 ms delay routine
;******************
d10_1ms
         movlw          .4                ;
         movwf          tempd1            ;
dly_1my  movlw          .204              ;
         movwf          tempd2            ;
dly_1mx  decfsz         tempd2,1   ;
         goto           dly_1mx           ;
         clrwdt                           ;
         decfsz         tempd1,1   ;
         goto           dly_1my           ;
         return                           ;

;******************
;10 ms delay routine
;******************
d10_10ms
         movlw          .32               ;
         movwf          tempd1            ;
dly_10y  movlw          .255              ;
         movwf          tempd2            ;
dly_10x  decfsz         tempd2,1   ;
         goto           dly_10x           ;
         clrwdt                           ;
         decfsz         tempd1,1   ;
         goto           dly_10y           ;
         return                           ;

        end
```

Recommended Assembly Flows

1.0 WAFER ON FRAME ASSEMBLY FLOW



2.0 WAFER ASSEMBLY FLOW



NOTES:

NOTES:

NOTES:

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
Microchip Technology Inc.
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-786-7200 Fax: 480-786-7277
Technical Support: 480-786-7627
Web Address: http://www.microchip.com

Atlanta
Microchip Technology Inc.
500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770-640-0034 Fax: 770-640-0307

Boston
Microchip Technology Inc.
5 Mount Royal Avenue
Marlborough, MA 01752
Tel: 508-480-9990 Fax: 508-480-8575

Chicago
Microchip Technology Inc.
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

Dallas
Microchip Technology Inc.
4570 Westgrove Drive, Suite 160
Addison, TX 75248
Tel: 972-818-7423 Fax: 972-818-2924

Dayton
Microchip Technology Inc.
Two Prestige Place, Suite 150
Miamisburg, OH 45342
Tel: 937-291-1654 Fax: 937-291-9175

Detroit
Microchip Technology Inc.
Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI 48334
Tel: 248-538-2250 Fax: 248-538-2260

Los Angeles
Microchip Technology Inc.
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888 Fax: 949-263-1338

New York
Microchip Technology Inc.
150 Motor Parkway, Suite 202
Hauppauge, NY 11788
Tel: 631-273-5305 Fax: 631-273-5335

San Jose
Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408-436-7950 Fax: 408-436-7955

AMERICAS (continued)

Toronto
Microchip Technology Inc.
5925 Airport Road, Suite 200
Mississauga, Ontario L4V 1W1, Canada
Tel: 905-405-6279 Fax: 905-405-6253

ASIA/PACIFIC

Hong Kong
Microchip Asia Pacific
Unit 2101, Tower 2
Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2-401-1200 Fax: 852-2-401-3431

Beijing
Microchip Technology, Beijing
Unit 915, 6 Chaoyangmen Bei Dajie
Dong Erhuan Road, Dongcheng District
New China Hong Kong Manhattan Building
Beijing 100027 PRC
Tel: 86-10-85282100 Fax: 86-10-85282104

India
Microchip Technology Inc.
India Liaison Office
No. 6, Legacy, Convent Road
Bangalore 560 025, India
Tel: 91-80-229-0061 Fax: 91-80-229-0062

Japan
Microchip Technology Intl. Inc.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa 222-0033 Japan
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

Korea
Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea
Tel: 82-2-554-7200 Fax: 82-2-558-5934

Shanghai
Microchip Technology
RM 406 Shanghai Golden Bridge Bldg.
2077 Yan'an Road West, Hong Qiao District
Shanghai, PRC 200335
Tel: 86-21-6275-5700 Fax: 86 21-6275-5060

ASIA/PACIFIC (continued)

Singapore
Microchip Technology Singapore Pte Ltd.
200 Middle Road
#07-02 Prime Centre
Singapore 188980
Tel: 65-334-8870 Fax: 65-334-8850

Taiwan, R.O.C
Microchip Technology Taiwan
10F-1C 207
Tung Hua North Road
Taipei, Taiwan, ROC
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

EUROPE

United Kingdom
Arizona Microchip Technology Ltd.
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44 118 921 5858 Fax: 44-118 921-5835

Denmark
Microchip Technology Denmark ApS
Regus Business Centre
Lautrup hoj 1-3
Ballerup DK-2750 Denmark
Tel: 45 4420 9895 Fax: 45 4420 9910

France
Arizona Microchip Technology SARL
Parc d'Activite du Moulin de Massy
43 Rue du Saule Trapu
Batiment A - ler Etage
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

Germany
Arizona Microchip Technology GmbH
Gustav-Heinemann-Ring 125
D-81739 München, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

Italy
Arizona Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-039-65791-1 Fax: 39-039-6899883

11/15/99

DNV Certification, Inc. USA — ANSI-RAB QMS ACCREDITED — DNV — DNV MSC The Netherlands Accredited by the RvA
ISO 9001 / QS-9000 REGISTERED FIRM

Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs and microperipheral products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001 certified.