THE GAME OF LIFE

The hardest part of writing this article was finding a way to describe what we're building. If you're already familiar with John Conway's "Game of Life" then you'll know instantly from the photo what the whole project is about, but if you're not, then what to say?

I could say that it's a simulation of a cellular automata invented some forty years ago by a British mathematician, but if I did you'd probably stop reading right now. I could also say that it's a super nifty cool

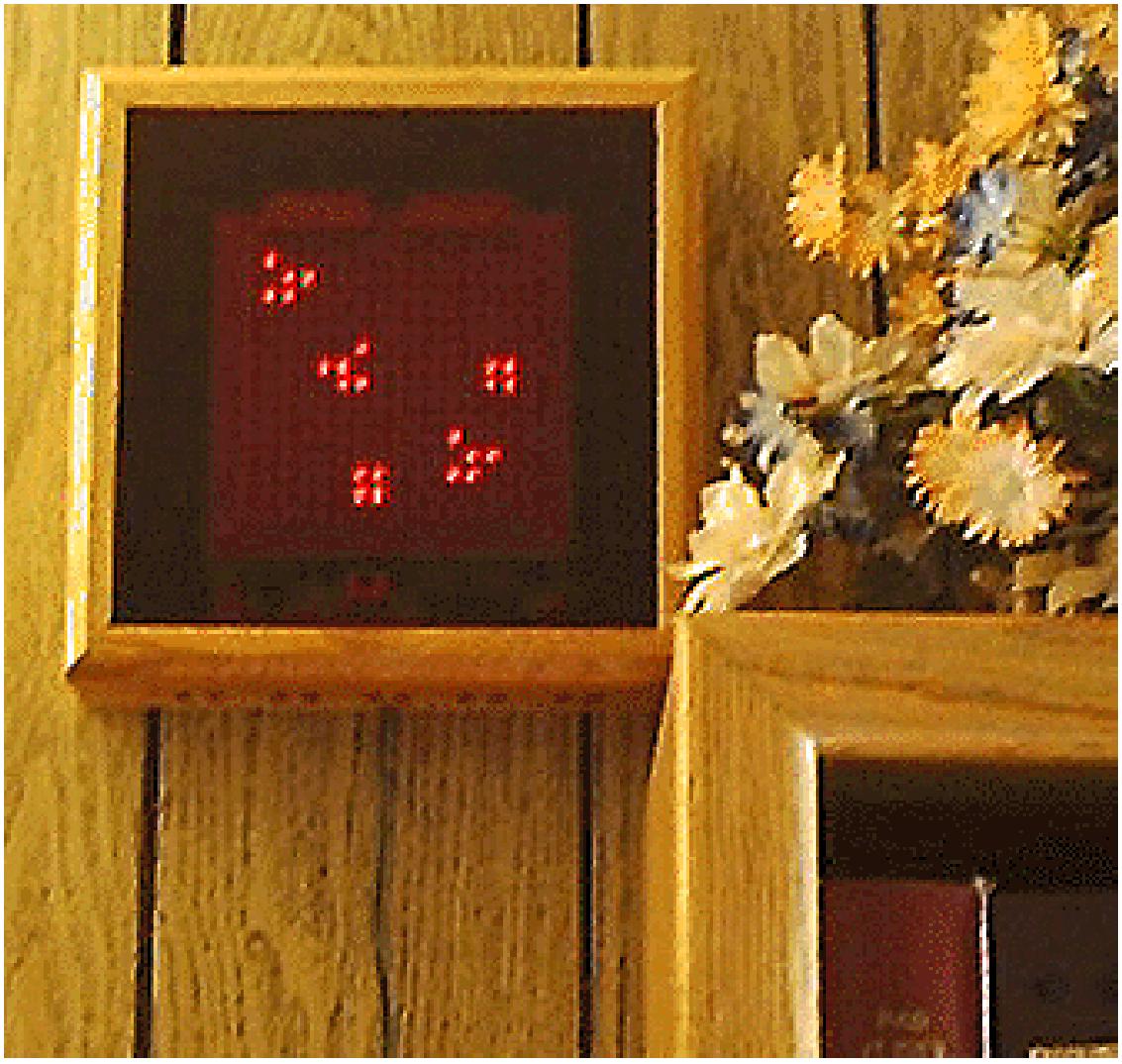


Photo 1 - Is it Art?

blinking light gizmo, but that doesn't do justice to the subtle complexity behind the scenes. Recently I've started telling people that "it's art, and a performance piece at that," and as such doesn't need any explanation. That's the original reason I created it, and that's the reason it hangs on the wall like a fine painting, and it's the explanation that works the best.

HARDWARE

Despite the apparent complexity - the complete schematic requires four pages - the hardware of the LIFE Game is actually pretty straight forward. The main part consists of four Maxim MAX7219 multiplexed display controller ICs. The 7219s are primarily used to control a row of seven segment numeric LED displays, however they also have a "no decode" mode which allows the software to have explicit control over every segment in the display. Used this way each 7219 can display an arbitrary pattern on an 8x8 patch of 64 LEDs.

The MAX7219s have several features which make them perfect for this application. First, each 7219 has its own internal oscillator, 64 bits of display RAM, and multiplexing logic. Once the software has downloaded a particular display pattern to the 7219, the chip can display and refresh that pattern on the LEDs without any further help from the software. This means that the software doesn't need to continuously refresh the display, which reduces the time demands on the microcontroller considerably.

Also, each 7219 has its own internal constant current source which is used to drive the LEDs. This ensures that each LED receives the same drive current regardless of any manufacturing variations from one LED to another, and improves the uniformity of the LED brightness. Better still, the 7219's constant current source is programmable, so that the software can actually change the brightness of the display. The current firmware uses this feature to do several nice fade-in/fade-out effects.

The 7219s use a simple serial data bus to receive data and commands from the microprocessor. The four chips in this project are daisy chained so that bits "flow thru" the first chips in the chain until they reach the last one - the software shifts out

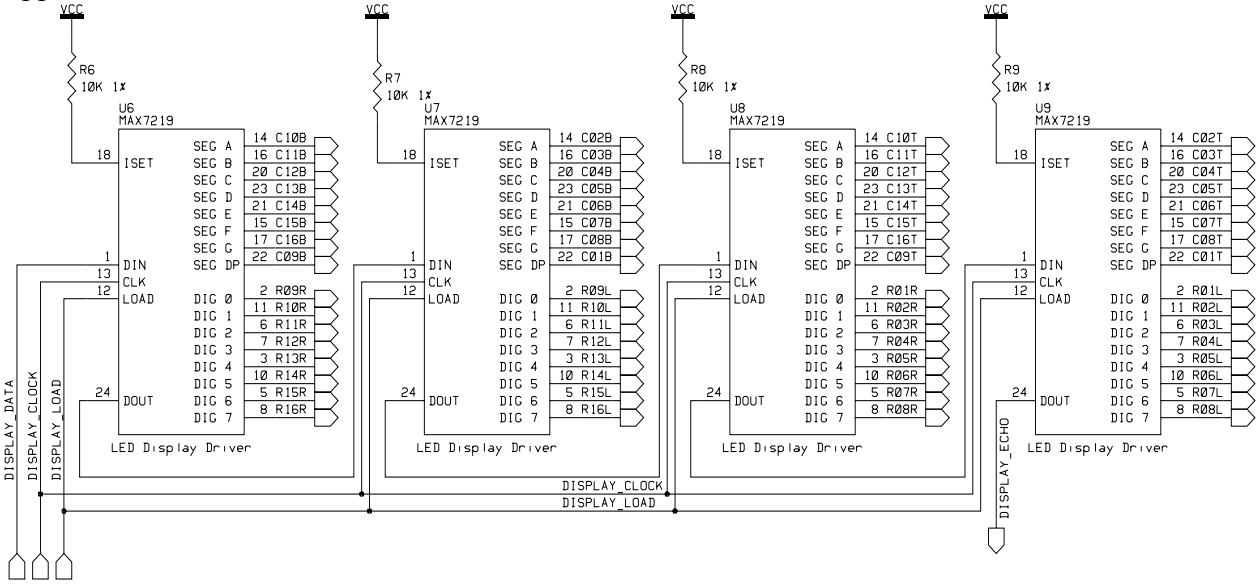


Figure 1 - Display Controller Wiring

the bits for U9 (the *last* one in the chain) *first* and the bits for U6 (*first* in the chain) *last*. The DISPLAY_DATA signal (port 1 bit 5 for the microprocessor) provides the serial data to the 7219 chain, and a rising pulse on the DISPLAY_CLOCK (port 1 bit 3) signal shifts each bit into the 7219. After all 7219s have received their bits, the software pulses the DISPLAY_LOAD signal (port 1 bit 4) which causes each 7219 to latch the data and act on it.

If you intend to build a LIFE game without a PC board, then wiring up all the LEDs and the outputs of the MAX7219s is quite an undertaking. The nomenclature used in the schematic is easy to understand, though. For example, the 7219 pin labeled C10T is a column driver for column 10 of the top section of the display. Likewise R04B would be a row driver for row 4 of the bottom display section. Rows and columns are numbered from 1 to 16 starting from the upper left corner of the display. IC U9 controls the upper left quadrant of the display (Rows 1..8 Left, Columns 1..8 Top); U8 controls the upper right quadrant; U7 controls the lower left, and finally U6 controls the lower right (rows 9..16 Right, Columns 9..16 Bottom). Figure 2 shows the wiring for just one quarter of the LEDs in the display.

Fortunately, a double sided silk screened and solder masked PC board is available from Spare Time Gizmos and makes wiring a snap.

A few last words on the 7219s - the resistors R6, R7, R8 and R9 program the constant current source in the chips. These should be 1% precision resistors to ensure uniform brightness in all four display quadrants, and the 10K value specified results in a maximum LED current of approximately 40mA. That's 40mA *per LED* so if a particular 7219 happens to have a column with all 8 LEDs lit, it would require 320mA total. And that's *per 7219* so the worst case power supply current required by the display would be 4x320mA or about 1.3A! Keep that in mind when we talk about the power supply section. Also, this is why the power connections for each 7219 are bypassed by both a 0.1uf ceramic capacitor *and* a 10uF tantalum - to prevent the high currents caused by switching LEDs on and off from causing spikes on the VCC supply.

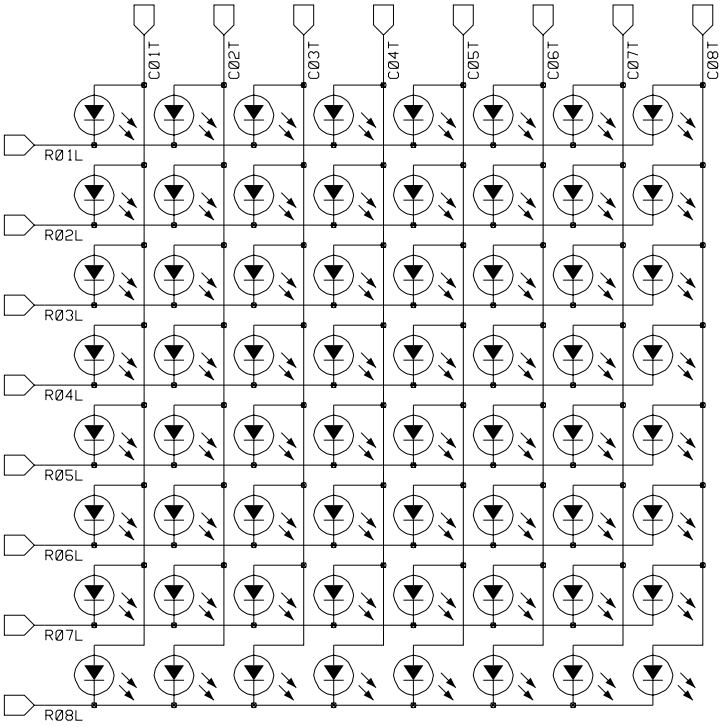


Figure 2 - LED Wiring (Upper Left Quadrant Only)

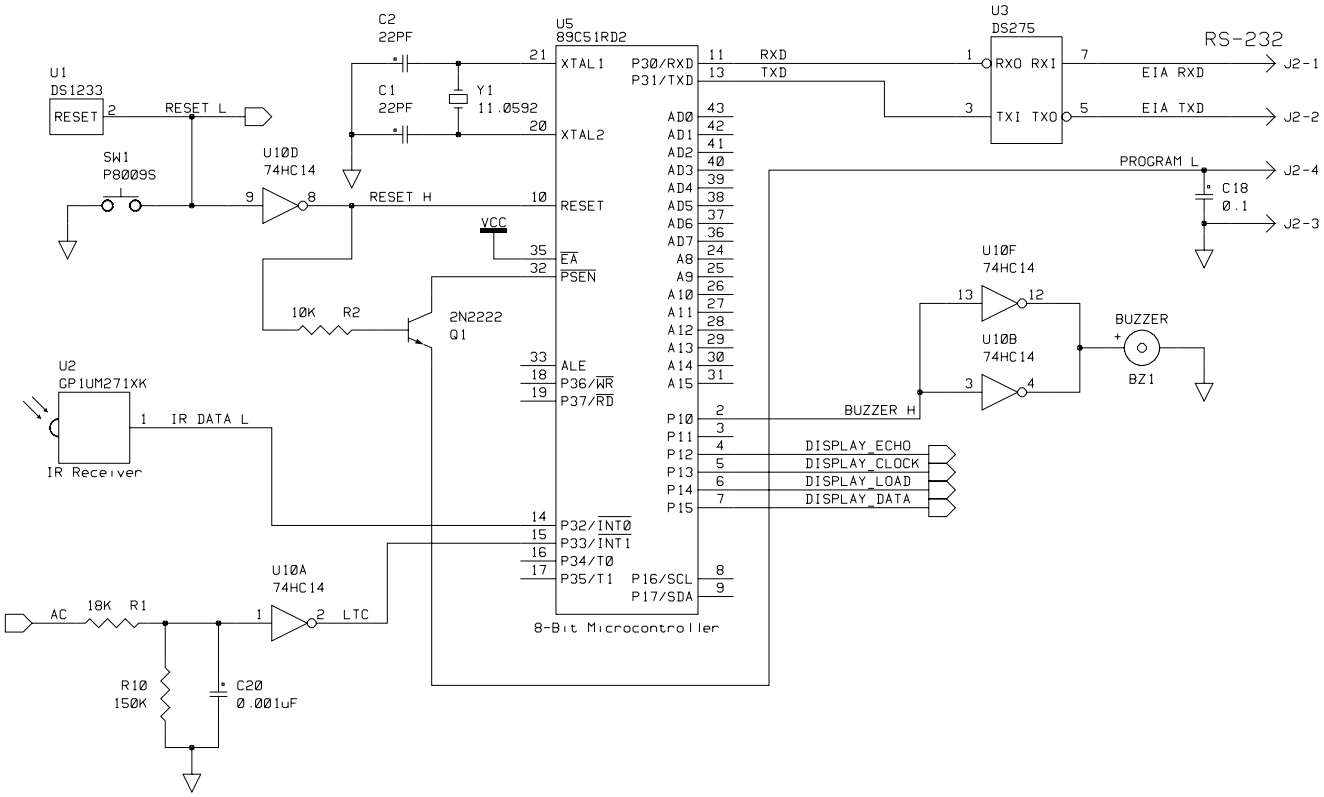


Figure 3 - Microprocessor Section

Microprocessor

The microprocessor used in the LIFE game is a modern variation of the venerable Intel 8051, which has been around for at least twenty years. The 8051 variant used here features flash memory for program storage and can be programmed, while it's still plugged into the LIFE game, from a PC using a standard RS-232 serial port. No special programmer is required! This is also why no EPROM is used in the design and it makes it easy for you to download the firmware to your game after it's built.

It also makes it easy to modify and change the firmware, and Spare Time Gizmos has made all the source code for the LIFE Game open source in hopes that people will add new patterns and features to the game.

Other than the flash memory and in system programming, the LIFE firmware needs no other special 8051 features (e.g. PWMs, counter/timers, A/Ds, I^2C or SPI ports, etc) and there are actually a number of different 8051 variants that can be used. The "preferred" variant is the Philips 89C51RA2, but this one is preferred only because it's the simplest and cheapest part that will run the LIFE firmware. The Philips 89C51RB2, RC2 and RD2 differ only in the amount of program memory they have (but all have more than the RA2) and any one of those will work.

Atmel also makes many 8051 derivatives with flash memory, however only a few of those support in system programming and only a few of *those* support programming with a RS-232 serial port. The Atmel AT89C51Rx2 series *will* work, and parts such as the AT89C51RB2 can be used. Finally, the Philips 89C66x series of parts will also work - these chips have extra features (e.g. I^2C, PWMs, etc) that we don't

need, but with a few trivial software modifications to disable these features they'll work fine.

The rest of the circuitry around the microprocessor is pretty conventional. The DS1233 provides a clean power on reset and a reset if VCC drops too low, plus it can be manually activated by a simple push button. Since the 8051 has an active HIGH reset input (pretty unusual these days!) we have to invert the output of the DS1233 with one section of the 74HC14. Transistor Q1, along with R2, pulls the processor PSEN pin low as long as reset is asserted - this little trick is unique to the flash memory versions of the 8051 and activates the built in boot loader. Note that the emitter of Q1 is grounded via the serial port cable, so that the boot loader will be activated *only* when the programming cable is connected. The Dallas DS275 is a super simple RS-232 EIA level shifter which allows us to interface the microprocessor's serial port to the PC with proper EIA signal levels.

Infrared Receiver

U2 is a Sharp GP1UM272XK infrared remote control receiver intended for a carrier frequency of 36kHz, the same frequency used by the RC5 remote control protocol. If you have trouble obtaining this part you can also use the GP1UM271XK – this part is has a carrier frequency of 38kHz which, while it is not ideal for RC5, in practice it works fine with perhaps a slight reduction in range. Be warned, though, that Sharp also makes several other versions with carrier frequencies even farther away than the 38kHz part, and these should be avoided. The serial data from the infrared receiver is applied directly to the INT0 microprocessor input.

Line Frequency Clock

A sample of 60Hz AC (or 50Hz, depending on where you live) is sampled from the power supply and applied to the input of Schmidt trigger U10C. If an 8VAC wall transformer is used then this input will receive a peak voltage of at least 12V, however the input protection diodes in the 74HC14 will clamp it to VCC. Resistor R1 is there to limit the current thru the clamp diodes to a safe value, and R10 and C20 filter the AC waveform and remove any high frequency noise. The output of U10C is a clean 60Hz square wave which is applied to the microprocessor's INT1 input, where the software uses it to keep track of the time of day.

Power Supply

The power supply is unusual because it uses an AC wall wart (a real transformer and nothing more) instead of the much more common DC output variety. This is done just so that we can derive the 60Hz line clock, but it also means that we have to include a bridge rectifier (D1) and filter capacitors (C3, C4, C5 and C19) on the PC board. Three or four medium value filter capacitors are used rather than one large one

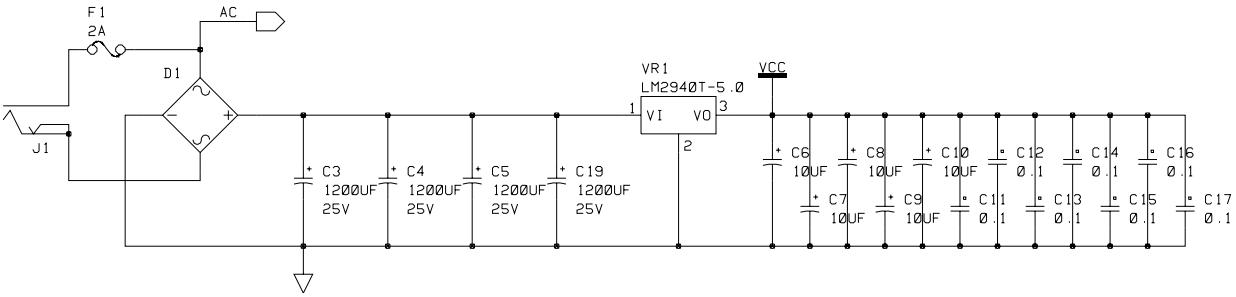


Figure 4 - Power Supply

strictly for mechanical reasons - a single large capacitor would be physically too big for the space available.

Although the power supply circuit is straight forward, the components are all carefully selected to minimize the heat dissipation of regulator VR1. In particular you should use the 8V AC wall wart specified in the parts list; any higher voltage one will make VR1 run hotter. *You cannot use the much more common 7805 regulator in this circuit*, either – the LM2490 was selected because it's able to work with as little as 0.5V difference between the input and output where as the 7805 requires at least 3V. Finally, 3,600uF of filter capacitance (C3, C4 and C5) is generally enough -- adding more only (C19) increases the dissipation of VR1 and you should do so only if you really need it.

With exactly the power supply components specified, VR1 will dissipate around 4 or 5 watts with all LEDs on. This is certainly enough to require a heat sink and good ventilation. But the reality is that the game only spends one second during the power on self test with all LEDs on, and the number of LEDs on at any instant while playing LIFE is only a small fraction of the total. Under these conditions VR1 can run forever with no heat sink other than the copper area provided on the PC board and it will barely even get warm.

The question of heat sinking VR1 is left up to the builder. If you only want to play LIFE, no heat sink is required. On the other hand, if you want to modify the firmware to display different patterns, especially ones with large numbers of LEDs on at one time, then you'll need a heat sink of some kind. Also be aware that some of the clock modes have higher LED duty cycles and will make VR1 run hotter. Remember that the LM2490 does have internal over temperature protection, and if it does overheat it'll simply shut down until it cools off.

SOFTWARE

The firmware for the LIFE Game consists of some eight thousand lines of 8051 assembly language code divided into twelve modules. Figure 5 shows the most significant modules and their relationship to each other and major hardware components. Execution is a simple foreground / background model with interrupt driven modules such as the RC5 (infrared) receiver and clock in the foreground and the life engine soaking up all the background time left over.

INFRARED.ASM

This module contains the code for receiving and decoding infrared remote control messages transmitted using the Philips RC5 code. RC5 codes are fourteen bits long with a nominal bit time of about 1.778 ms. Most remotes use nothing more than a cheap R/C oscillator for a time base, though, and you can't depend very much on the exact timing. Instead, RC5 uses a biphase encoding system where every bit is self clocking (in this case, every bit contains a transition in the middle of the bit time window) and the state of the bit (a one or a zero) depends on the polarity of that transition.

DISPLAY.ASM

The code maintains a bitmap of the state of the 256 LEDs in an internal 32 byte buffer in the module RAM.ASM. The DISPLAY.ASM module contains the code to send this internal bitmap, along with the necessary commands, to the four Maxim 7219 chips. Remember that the MAX7219s are self refreshing, and so the DISPLAY module is invoked only when the bitmap actually changes (i.e. the next generation is computed). Once the bitmap has been

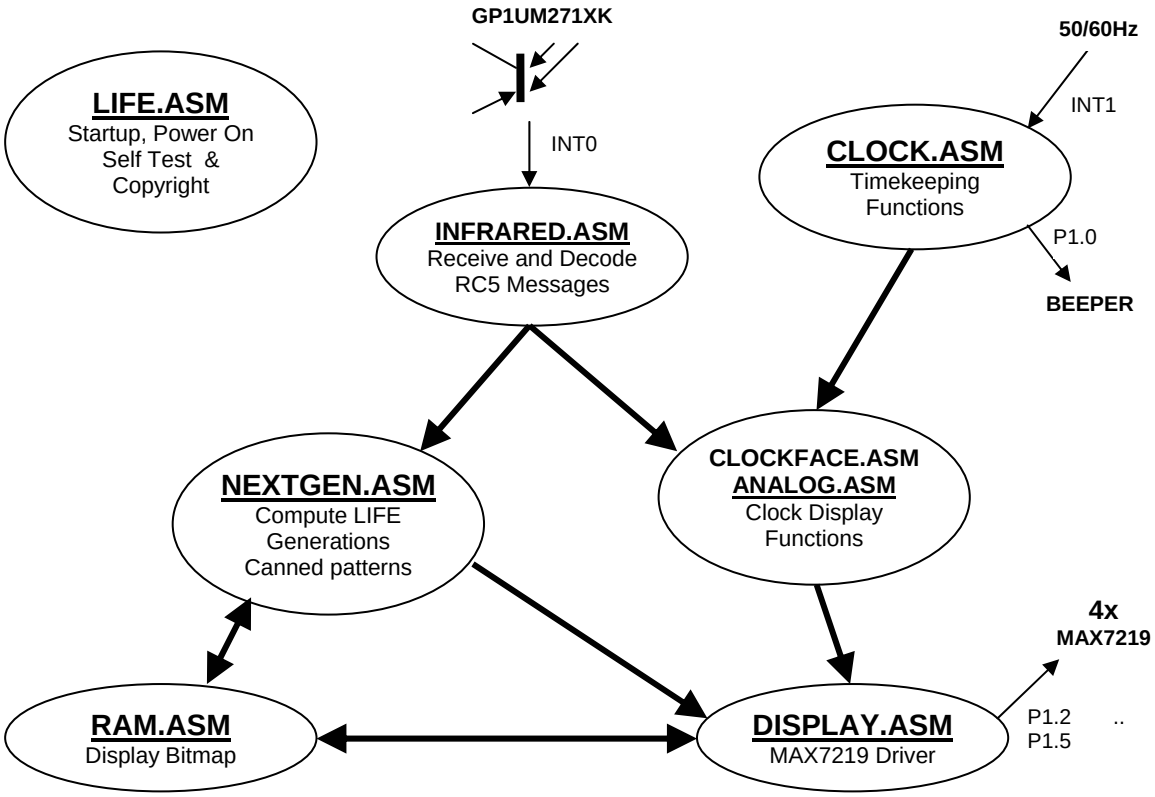


Figure 5 - Software Architecture

loaded into the 7219s, they can maintain a static display without any further help from the software.

NEXTGEN.ASM

The NEXTGEN module contains the actual LIFE "engine" - the function which computes the next LIFE generation from the current one. Doing this on the 8051 required a few "tricks" to conserve RAM since the 8051 has only 128 *bytes* of RAM and half of that's taken up by the registers, display buffer and stack space. Interestingly, although the 8051 is short on RAM, it has plenty of ROM for this application and the algorithm used contains few loops - they've all been "unrolled" to optimize execution speed at the cost of ROM space. Even on a lowly 11MHz 8051 the NEXTGEN code can compute life generations faster than you can see them, which is all we need for this application!

CLOCKFACE.ASM ANALOG.ASM

These modules display the time in various formats, with and with out LIFE effects. ANALOG.ASM displays a simulation of an actual analog clock, with hours, minutes and seconds hands (by a simply amazing stroke of luck, a 16x16 display has *exactly* 60 LEDs around the perimeter!). CLOCKFACE.ASM displays a digital clock simulation with the hours and minutes displayed in a 4x5 numeric font. Either clock option can play LIFE with the current clock display.

CLOCK.ASM

This module counts the 50/60Hz line time interrupts and keeps track of the actual time of day and date. The current time can be set via the remote control, and is used by both the CLOCKFACE.ASM and ANALOG.ASM modules to display the time. This module also contains the code to turn the beeper on and off, which belongs here only because it uses the line clock to time the duration of the beeps...

LIFE.ASM

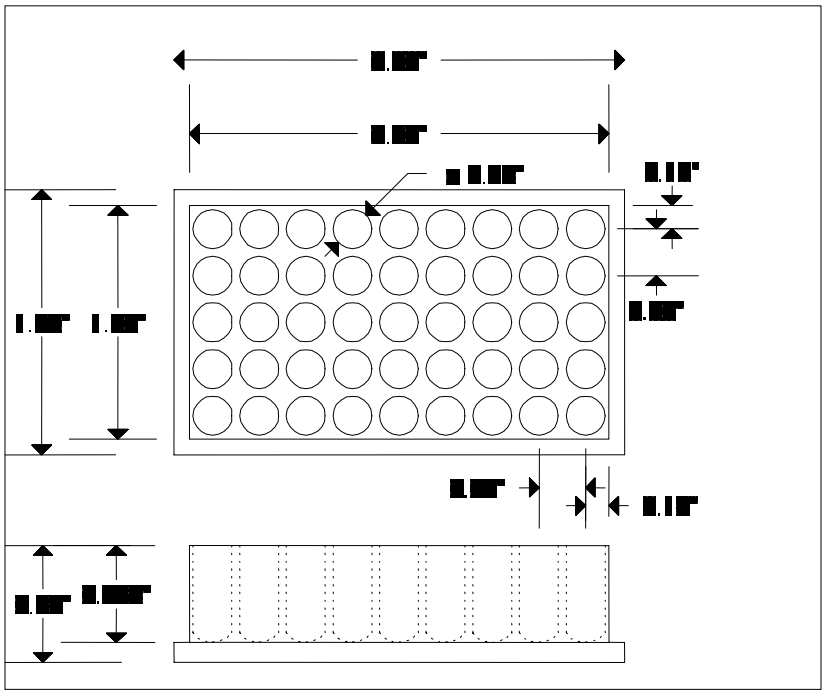
This is the "main module" of the firmware. It contains the startup and initialization code as well as the interrupt vectors. It also contains a power on self test (aka POST) function which executes a simple test of the hardware whenever the system is powered up. Although it's by no means exhaustive, the POST does give the builder some confidence that things are working correctly.

CONSTRUCTION

The thought of soldering 256 LEDs may sound a little daunting, but it's not all that bad. I've built several and I can generally do the LED part in about an hour and a half. The single most important factor in building a good looking display is that the LEDs be uniformly spaced – this is easy to do if you use a special jig for installing them.

The dimensions of a suitable LED jig are shown in Figure 6 and Photo 2 is what a finished LED jig looks like. The jig shown in the photo is made of aluminum, which is easy to machine but probably a little extravagant for our needs. PVC or ABS plastic, or even wood, could also be used. The number of rows and columns in the jig aren't critical and you can make yours larger or smaller; the only dimensions that you absolutely must have correct are the LED spacing of 0.300" and the LED depth of ⅝" (0.625").

Figure 6 - LED Alignment Jig



If you don't want to make your own LED jig, you can buy one from Spare Time Gizmos. The ordering information is given at the end of the parts list. Since the LED jigs are admittedly expensive for something you use only once, Spare Time Gizmos will offer a partial refund for every LED jig that is returned.

The jig is easiest to use if you install the LEDs first, before there are any other parts on the PCB that might interfere with it. Start by inserting a block of 45 LEDs into the PC board (assuming your jig is 5 x 9) and then fit the alignment jig on top of them. When inserting the LEDs be sure to watch the polarity – the LED has both a long lead, which goes into the hole marked with a "+" sign, and a flat spot

on the side of the case, which is shown on the PCB silk screen.

After inserting the LEDs into the PC board and the jig, flip the entire assembly over and rest the jig on your workbench with the PC board on top. You'll need to use something to prop up the far end of the PC board so that it rests level. Carefully tap each LED's leads to make sure that the LED has fallen all the way down and that the face of the LED rests against the bottom of the hole in the jig. Solder all the leads, and while you're soldering take the opportunity to double check that each LED is installed properly by making sure that it's long leg is on the same side as all its neighbors.

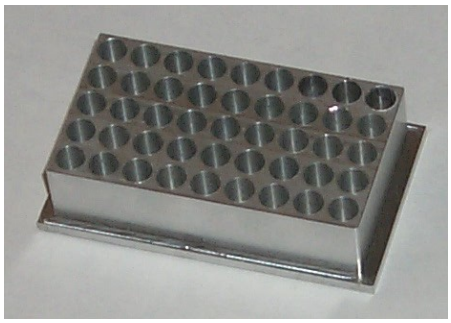


Photo 2 - LED Alignment Jig

Flip the board over, pop off the jig, and you should be rewarded with a block of 45 perfectly spaced LEDs. Repeat the entire process with another block of LEDs adjacent to this one, but this time *be sure to overlap at least one row or column of already soldered LEDs* in the jig. This ensures that each new section of LEDs is consistently spaced compared with the ones already installed.

Some of you are undoubtedly wondering why I didn't use four pre-assembled 8x8 LED matrix displays instead of discrete LEDs. The simple answer is that, although 5x7 matrix displays are very common, the 8x8 variety is not and I was unable to find any supplier that regularly stocked and sold suitable displays in single piece quantities. Yes, 8x8 LED displays are often available from surplus parts vendors, but with surplus vendors there's no guarantee that they'll have the same parts available tomorrow. That's not much good for publishing a construction project!

A word on LED selection – the "super bright" high output red LEDs make the brightest display, but you can use other color LEDs if you like. The photo shows a display made from green LEDs which looks quite nice, and you should also be able to use yellow or orange LEDs without any problem. Blue and White LEDs are a different story, however – most of these will have voltage and current requirements which are beyond the ability of the MAX7219s.

Finishing the PC board

After you have the LEDs installed set the PC board aside and take a break for a few minutes. The hard part is over and finishing the rest of the PC board should be a snap.

Remember that you must use the LM2940T-5.0 regulator for VR1 – don't substitute the more common 7805! Also, don't confuse the LM2940T (no suffix), which is an adjustable output regulator, with the LM2940T-5.0 version, which is fixed at 5V. Whether you want a heat sink for the regulator is up to you; refer to the Power Supply section for a discussion of this question.

Capacitor C19 is optional and shouldn't be installed unless you need it; this will be discussed later in the Checkout section.

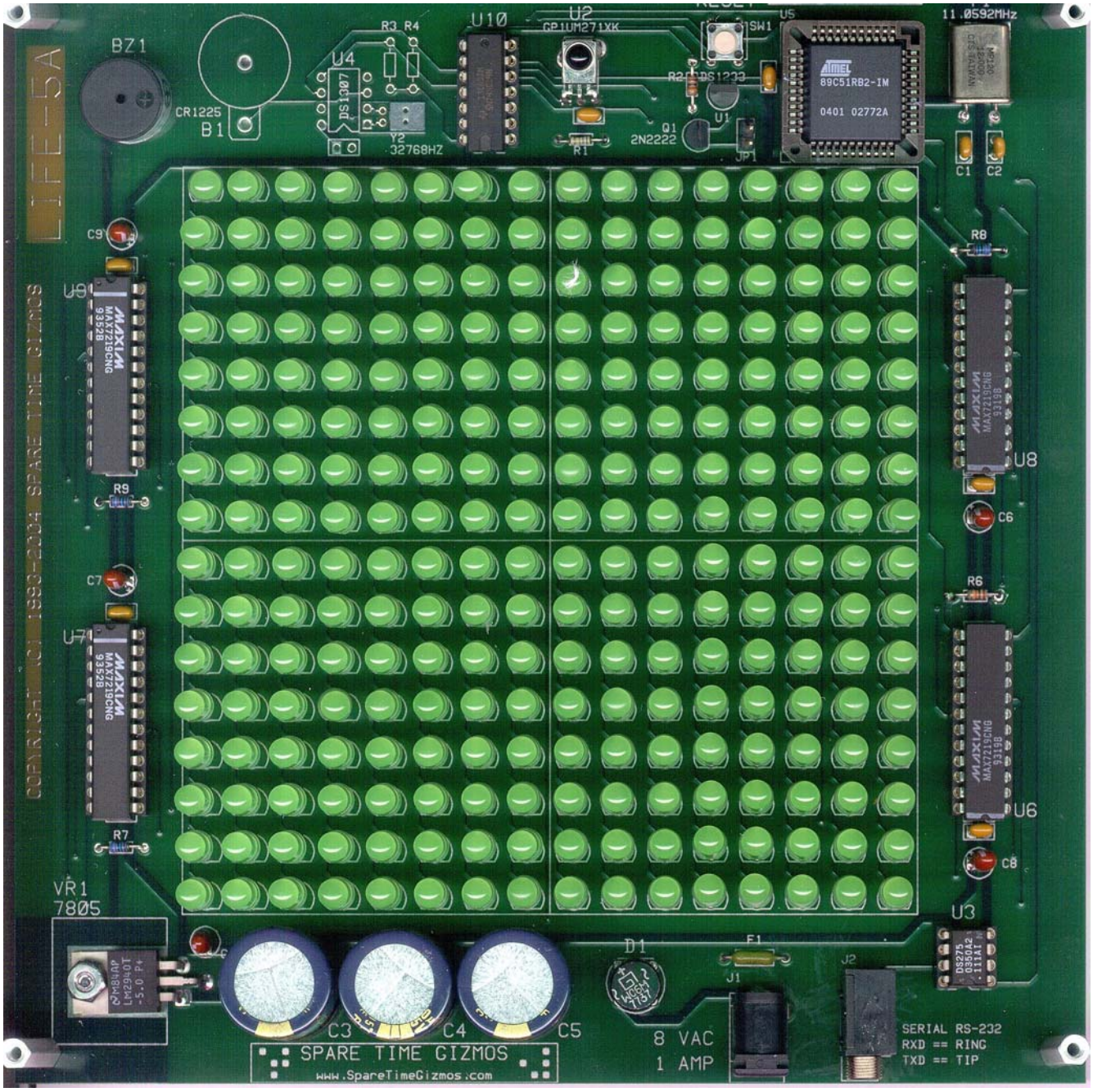


Photo 3 - Assembled LIFE Game PC Board

Components R3, R4, Y2, B1 and U4 in the upper left corner are for a future expansion project and are not used in the basic version of the LIFE Game. None of these parts should be installed.

If you elect to purchase a preprogrammed microprocessor and have no desire to update your firmware in the future, then you may

omit parts SW1, R2, Q1, U3 and J2.

After you've finished construction but before you install any ICs, apply power and check for +5V at the output of VR1. If this test is successful, install the remaining ICs and proceed with programming your microprocessor.

PROGRAMMING

However you choose to do it, your microprocessor must be programmed before it can be used. If you purchased a preprogrammed microprocessor from Spare Time Gizmos, or if you have access to a conventional EPROM programmer and want to use it to program your microprocessor, then you can skip this section.

The wonderful thing about the Philips and Atmel microprocessors used in this project is that they can be programmed by any PC with nothing more than a standard COM port. No special programmer hardware is required. Better yet, the microprocessor can be erased and reprogrammed with out any special UV eraser, and, without even removing it from the PC board!

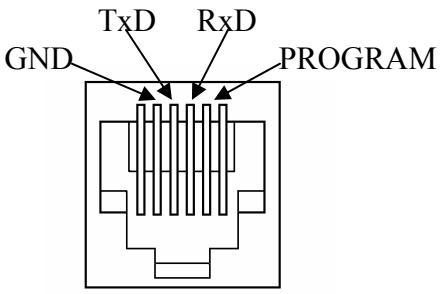


Figure 7 - Programming Connector

Before you can proceed you need to make up a programming cable to connect the RJ11 serial port on the LIFE Game to the DB-9F or DB-25F serial port connector on your PC. Most electronics stores stock DB-9 and DB-25 to RJ11 adapters – these come unfinished and you wire them yourself by inserting the pins and wires in the appropriate holes for the DB connector. If you have of these together with a standard RJ11 telephone cord you can make short work of the programming cable.

If you do use a telephone cord, examine it very, very carefully before you use it. A standard telephone cable *reverses* the order of the pins and your cable won't work unless you take this into account. A cable intended for data rather than telephone use doesn't ordinarily reverse the connections.

Figure 7 shows the pin out of the RJ11 connector on the LIFE game, and Table 1 shows the corresponding PC pins for both a DB-9F and DB-25F connector. Note that only three signal from the LIFE game are connected to the PC – Ground, TxD and RxD. The fourth pin, PROGRAM, *should be connected to GND* – this enables the microprocessor programming mode whenever the RJ11 connector is plugged into the LIFE game. Finally, the connector called for in the parts list actually has six pins, but only the innermost four pins are used.

	DB-9F	**DB-25F**
GND	Pin 7	Pin 5
RxD	Pin 2	Pin 3
TxD	Pin 3	Pin 2

Table 1 - PC Programming Connections

Programming Philips MCUs

To program the Philips 89C51RB2, RC2 and RD2 microcontrollers, visit the Philips Semiconductors web site

http://www.semiconductors.philips.com/markets/mms/products/microcontrollers/support/software_download/8051/index.html

and download the WinISP file. Unpack the file and follow the instructions to install it on your PC.

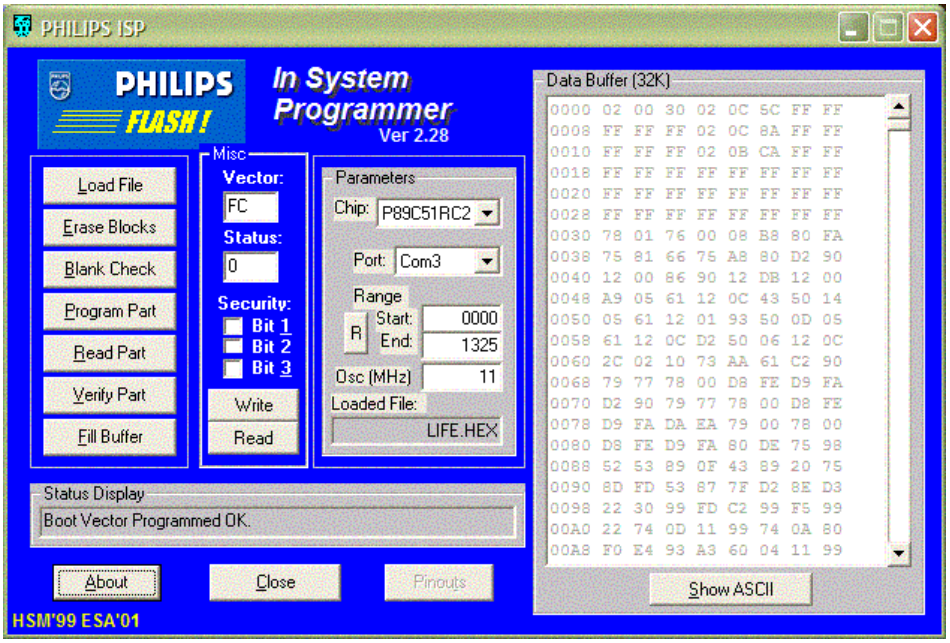


Photo 4 - WinISP Screen

Disconnect power from the LIFE Game and connect one end of the programming cable to your PC and the other end to the LIFE Game. On your PC, run the WinISP program.

In WinISP, click the *Chip* drop down list and select the specific P89C51 chip that you're using. Click the *Port* drop down list and select the COM port. In the OSC (Oscillator) box, enter *11* for a 11.0592MHz crystal (only enter the integer part of the crystal frequency – it won't accept a decimal value!).

Apply power to the LIFE board and, back in WinISP, under the *Misc* section, click the *Read* (Vector) button. If you see the message *Boot Vector Read OK* in the status window then all is well; if you get any errors then stop to figure out what's wrong before proceeding.

Click the *Load File* button and browse to the HEX file for the LIFE Game firmware. Click the *Erase Blocks* button and then, in the popup window, click *Full Chip Erase*. Answer *Yes* to the warning window that you'll get. Wait for the erase chip to finish – it takes a minute or so.

Next click *Blank Check* to verify that the part is erased, and then click *Program Part*. If all is well you should see the message *Flash Programming Successful* appear in the status window. If you like, you can click the *Verify Part* button to double check.

Finally, in the *Misc* window, change the *Status* byte to *0*. Don't change the *Vector* byte – it should be *FC*. Be sure that all three security options are *unchecked* and click the *Write* button. Answer *Yes* to the warning popup.

IMPORTANT!

If you do not change the status byte to zero, the microprocessor will not run the firmware you've downloaded! Don't forget this step. And never, ever, change the Vector byte to anything other than FC – if you do, you'll be unable to access the in system programming feature of the microprocessor in the future!

Programming Atmel MCUs

The procedure for programming the Atmel microprocessors is similar, but you cannot use the Philips WinISP software to program the Atmel parts. Instead, visit the Atmel 8051 tools web site:

http://www.atmel.com/dyn/products/tools.asp?family_id=604

and download the FLIP ("*FLexible In-*

system Programmer") program. Follow the instructions for installing FLIP on your PC.

Disconnect power from the LIFE Game and connect one end of the programming cable to your PC and the other end to the LIFE Game. On your PC, run the FLIP program and click the *Device >> Select* menu choices. Pick your particular AT89C51Rx2 chip from the list and click *OK*.

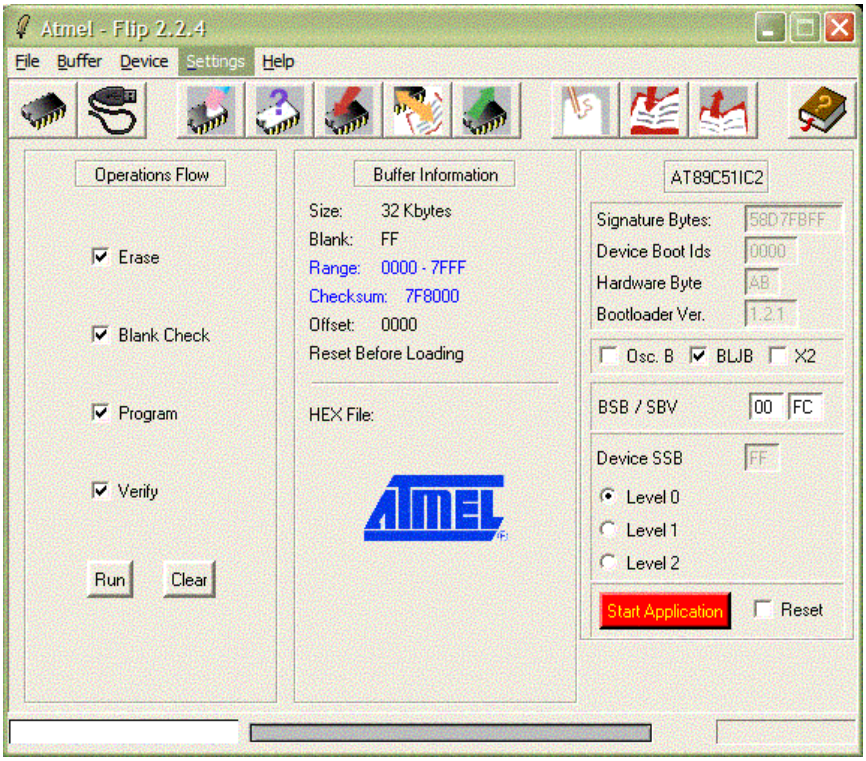


Photo 5 - FLIP Screen

Next click *Settings >> Communications* and *RS-232* on the menu and select the particular COM port that you're using. Pick 9600 as the baud rate. Apply power to the LIFE board and click the *Connect* button.

Pick *File >> Load Hex File* and then browse to the HEX file for the LIFE Game firmware. In the *Operations Flow* section of the main window, be sure that all four steps, *Erase*, *Blank Check*, *Program* and *Verify*, are selected.

Finally, click the *RUN* button and watch the status bar. You should see the tool erase the AT89C51, download the firmware and verify it. If you get any errors, figure out what's wrong before proceeding.

By the way, the Atmel tool, FLIP, is available in both Windows and Linux versions, but to the best of my knowledge the Philips WinISP tool is Windows only.

CHECKOUT

After you've successfully programmed the microprocessor, disconnect the power and programming cables. Plug in the power again and hold your breath!

The MAX7219s will reset to an all LEDs off state, and approximately 1 second later, if the firmware is alive, it will turn all LEDs on for about a second, followed by a short beep from the buzzer. After that, the copyright notice is displayed and the game will start playing LIFE. If you get this result then all is well and you can proceed to setting up your remote control!

If nothing at all happens then the firmware either failed to load properly or you forgot to disconnect the programming cable. Remember that the programming cable contains a jumper which grounds the PROGRAM signal – this places the microprocessor in program mode any time the cable is connected to the LIFE game.

If the device appears to hang in the "all LEDs on" state, then your power supply is inadequate. Refer to the Power Supply Problems section for more discussion. If an

individual LED does not light during the all LEDs on test, then it's either poorly soldered or a defective LED. An LED that stays lit all the time is most likely installed backwards, and if an entire row or column doesn't light correctly then you're looking for a solder bridge.

The firmware contains built in power on self test (POST) that executes at startup. If the POST detects a failure, it will use the buzzer to beep a code of one or more short beeps, pause, and then repeat the code. This continues until power is removed. Table 2 shows the POST error codes in the current firmware.

Beeps	Failure
1	ROM checksum failure
2	CPU failure
3	50/60Hz clock failure
4	MAX7219 failure
5	GP1UM272XK failure

Table 2 - POST Codes

Power Supply Problems

You'll know you need it if the game hangs during the "all LEDs on" part of the self test. All LEDs on is the worst case for the power supply and if the output of VR1 falls below about 4.5V the DS1233 will hold the microprocessor in the RESET state, leaving the display "frozen" with all LEDs on. If you do have this problem, then be sure not to leave the game this way (with all LEDS on) for extended periods - VR1 will heat up pretty fast under these conditions.

Some parts substitutions that can cause this problem are:

- using a wall wart rated for less than 8V and 1A
- using a regulator for VR1 other than the LM2940T-5.0
- using the 5% version of the DS1233 rather than the 10% model
- changing the MAX7219 current programming resistors

If you haven't done any of these things and you're still having problems, it may be that your line voltage a little low. Installing an additional filter capacitor, C19, in the power supply should correct the problem.

REMOTE CONTROL

Now that your LIFE game is working, you can go looking for a suitable place of honor to hang it. Once you've hung it on the wall, there's no worry about reaching it - you can operate the LIFE game from the comfort of your favorite chair using a conventional infrared remote control! The LIFE game firmware is able to decode the Philips RC5 remote control protocol and the game responds to the same command codes used by Philips' televisions.

Photo 6 - URC-4605

Photo 6 shows a One For All (www.oneforall.com) "Access 4" model URC-4605 universal remote. These are available from department stores, drug stores and hardware stores for less than $10. This control works perfectly with the LIFE Game, however pretty much any model universal remote that can be programmed for a Philips television could be used instead. Consult the instructions for your remote to find the correct setup code for a Philips television - in the case of the

URC-4605 it is #0054 (#1354 and #1254 also work).

The URC-4605 also has a feature that allows you to reassign device keys so that you can control more than one television by, for example, reassigning the DVD key to a second TV. This lets you control both your LIFE game and a real TV (as well as a VCR and a cable/satellite box) with a single remote!

When you have your universal remote set up, Table 3 shows the TV keys and their functions in the LIFE game.

Key	LIFE Mode	Clock Mode
MUTE	Next Mode	
POWER	Adjust Brightness	
MENU	reserved	
SLEEP	Auto Pattern Change	Freeze Clock
VOL +	Next Pattern	Set Hours
CH UP	Run Faster	Set Minutes
CH DN	Run Slower	Set Seconds
PREV	Show Copyright Screen	

Table 3 - Remote Control Keys

USING LIFE

The LIFE Game firmware has two distinct modes, LIFE and Clock, each with several sub modes.

LIFE Demo Mode

LIFE Demo mode cycles between the various interesting LIFE patterns built into the firmware. Each pattern runs for several generations, and then the game pauses and moves on to the next one.

Digital Clock Mode

This mode displays the current time of days, hours and minutes, as decimal digits, surrounded by a ring of 60 LEDs which light up one by one to indicate the seconds. This mode does *not* play LIFE.

Analog Clock Mode

This mode displays three concentric rings of illuminated LEDs to indicate the hours, minutes and seconds. This mode does *not* play LIFE.

Clock-Life

This mode is similar to the Digital Clock Mode except that after displaying the time for a few seconds, the game begins playing LIFE with the font for the hours and minutes digits. After playing LIFE for a while it returns to the digital time display and starts over again. This mode does not display the seconds.

Life-Clock

This mode starts off playing LIFE and, after a few seconds and a dozen generations or so, the current digital time will coalesce out of the LIFE pattern. It's effectively Clock-Life in reverse! I leave it as an exercise for the reader to study the display and discover how this little trick is accomplished (and no fair reading the source code, either!).

(Pseudo) Random Bit Stream

This mode displays a continuously flowing stream of pseudo-random bits. It doesn't tell the time and it doesn't play LIFE.

OFF

In this mode the display is blanked, however the microprocessor continues to run and keep track of the time of day and respond to keys on the remote.

Operation

The LIFE Game always powers up into LIFE Demo Mode. This is it's most interesting and useful mode and, if you don't have a remote control, there's no reason why

you can't simply leave it locked in that mode forever.

If you do have a remote control, pressing the *Next Mode* key (*MUTE* on the universal remote) rolls between all the modes. In the LIFE game modes, the *Faster* and *Slower* keys (*CH UP* and *CH DN* on the universal remote) change the speed at which the LIFE generations pass. In LIFE Demo Mode only, the *Auto Pattern Change* key (*SLEEP* on the universal remote) can be used to lock the demo on the current LIFE pattern only. In this case the *Next Pattern* key (*VOL +* on the universal remote) advances to the next built in LIFE pattern.

In the Digital Clock mode only, the *Freeze Clock* (*SLEEP*) , *Set Hours* (*VOL +*), *Set Minutes* (*CH UP*) and *Set Seconds* (*CH DN*) keys can be used to set the current time of day.

In any mode, the *Brightness* key (*POWER* on the universal remote) will change the display brightness and the *Show Copyright* key (*PREV* on the universal remote) will return to the Copyright screen.

PARTS LIST

Resistors

R1 – 18K 1/8W 5%
R2 – 10K 1/8W 5%
R3, **R4**, **R5** – unused
R6, **R7**, **R8**, **R9** – 10K 1/8W 1%
R10 – 150K 1/8W 5%

Capacitors

C1, **C2** – 22pF ceramic monolithic capacitor
C3, **C4**, **C5** – 1,200UF 25V radial lead aluminum electrolytic (Panasonic EEU-FC1E122S – Digi-Key P/N P10280)
C6, **C7**, **C8**, **C9**, **C10** – 10uF 6V radial lead tantalum capacitor
C11, **C12**, **C13**, **C14**, **C15**, **C16**, **C17**, **C18** – 0.1uF monolithic ceramic bypass capacitor
C19 – optional (see text) – same as C3 if used
C20 – 0.001uF ceramic monolithic capacitor

Semiconductors

LEDs – 256 Red T1-3/4 hi efficiency diffused LED (Kingbright L53ID – Mouser 604-L53ID)
D1 – PF05 1.5A 50PIV bridge rectifier
VR1 – National LM2940T-5.0 fixed output low dropout 5V TO-220 regulator
U1 – Dallas DS1233-10 microprocessor reset (Digi-Key DS1233-10)
U2 – Sharp GP1UM272XK 36kHz infrared receiver (Digi-Key 425-1142)
U3 – Dallas DS275 RS-232 single line receiver/transmitter (Digi-Key DS275)
U4 – unused
U5 – Atmel or Philips MCU (see text) (Digi-Key AT89C51RB2-SLSIM)
U6, **U7**, **U8**, **U9** – Maxim MAX7219 LED Display Driver
U10 – 74HC14 Hex Inverting Schmitt Trigger
Q1 – 2N2222 NPN GP transistor TO-92

Miscellaneous

Y1 – 11.0592Mhz HC-49 microprocessor crystal
Y2 – unused
B1 – unused
SW1 – PCB mount push button switch (Panasonic EVQ-PAD05R Digi-Key P10890S)
BZ1 – 3-16VDC piezo buzzer (Radio Shack 273-074)
J1 – high current coaxial power jack (2.1 x 5.5mm) (Cui-Stack PJ-102AH Digi-Key CP-102AH)
J2 – low profile RJ11 phone jack (CorCom RJ11-6N-B – Digi-Key CCM1394)
F1 – 2A slow blow pico fuse
– Silicore SLA40810 8VAC 1A wall transformer (2.1 x 5.5mm plug) (Jameco 219660)

Hardware

- PLCC44 IC socket
- DIP14 IC socket
- 4xDIP24 0.3" IC socket
- DIP8 IC socket
- #4-40 mounting hardware for VR1
- Heatsink for VR1 (optional - see text)

Other

The following special parts are available from Spare Time Gizmos:

- A ready to use PC board, with silk screen, solder mask and plated holes
- The special LED alignment jig
- A wooden enclosure kit similar to the one shown in the photographs, including plastic front bezel and a rear panel
- A full kit of all parts needed to build the LIFE Game

For more information about the LIFE Game, including prices and ordering information, visit the LIFE Game web site at

http://life.SpareTimeGizmos.com

or send email to

orders@sparetimegizmos.com.

ACKNOWLEDGEMENTS

A special thanks to Dan Liddell, who spent endless hours at the kitchen table with salt shakers and beer bottles, figuring out the optimal way to move bits around in the 8051 LIFE engine.

Thanks also to Philip Freidin and Fliptronics, www.fliptronics.com, whose CNC milling machine made the LED jig an easy job.