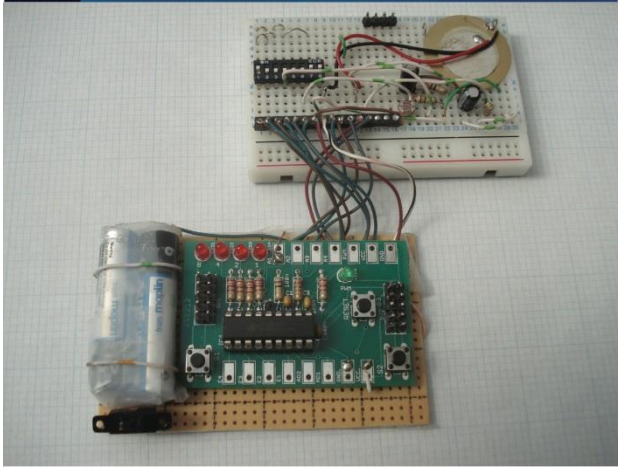
PDF Version just showing the Pics, Drawings and Code

Learning Programming with MyCo

By Juergen Pintaske

Easy to learn and PC independent – only this kit picture is in German



Contents

First I have to say thank you to the many people who helped to make this eBook happen, and just to mention a few:

Burkard Kainka, the genius behind the German kit who started me on this venture, which proved a lot more time consuming that expected.

Franzis Electronic Kits who sent me some kits immediately, so I could take my own photographs and to let others play with the kit and give feedback.

And my family:

PPP - Patrick Pintaske Photography helped with some of the pictures and editing

Lisa Pintaske with art help and advice

My wife Barbara, the one to help, check the book, correct and add her own inputs.

Ralf Lieb and Michael Schwope with feedback and inputs for future extensions, e.g. the build at the end

---

We have taken great care to ensure that all of the drawings are correct. We appreatiate feedback to epldfpga@aol.com to enable us to correct further editions of this eBook.

This eBook is the description of an existing kit as seen on the cover page. We denounce any liability regarding build and use of it, or for damages that might arise when used in applications. This is an educational device to be used as is, connecting it to aditional external components could be dangerous.

This is one of the 4 eBooks I have published recently:
Forth – The Early Years - goo.gl/y2Zlud
Forth – Programming a Problem Oriented Language - goo.gl/SVRdyF
Forth Tutorial using free MPE VFX Forth - http://goo.gl/7nK36V

0 – Background to this eBook

When I saw this little kit on the Internet and read what it can do, I could not believe it. A complete computer that you can program, including keyboard and display, input and output. No PC required. Yes, the absolute minimum, but it works. Looking at the low cost and having a bit of fun, was definitely worth the time and the money. You will have to solder the pieces together, but there should always be somebody around who can help if needed. And I assume soon you will be able to buy and sell the kits on eBay already soldered together – ready to go.

If you look at pictures 1, 2 and 3 on the cover page, you see the original box it came in and the little PCB with the Microcontroller on it. There are additional parts included to start you off with the first experiments using pre-programmed code. No need to learn programming first. To make life easier for me afterwards, I soldered wires on to the relevant pins while doing the soldering, to be prepared for later experiments. In this way the hardware would be ready, I just plug this additional new extension connector into the breadboard, and add the components for the experiment - no more soldering.

And the kit worked first time. A wonderful learning toy, I assume the age range could be about 5 to 95. Having gone through the examples, you will understand the basic workings and structure of a computer and as well you can do a little of your own programming. And kids can use it to add functions to their toys – all under software control. I will soon give the kit to some of my neighbors who are teachers, asking for feedback about what the kids think having played with it. As I live in the UK now, it all has to be in English for them, but this kit is only available in German at the moment. So I decided to write my first eBook.

To give the English speaking community the chance to relate to the German booklet that comes with the kit, I tried to keep the sequence the same. The name of the kit in German is TPS (roughly translated Switch Programming System), but the designer Burkhard Kainka allowed me to give it my own name. There was a short christening, and MyCo was born - My little Computer.

I hope you have fun reading this eBook, and if you are brave, you might even order the German Kit - I got mine via Amazon within 3 days, build it and enjoy. All of the information in the German booklet and more is in this eBook. To help with your first programming even without having the kit, I generated the Programming Page in the Appendix. Fill in your program and see how the data flows, writing code and data into the relevant boxes. All of the programs in this eBook are identical. Even many of the pictures. If some of the information seems to be too difficult, you can continue with the examples and go back later.

There is a lot more that has happened around this little kit in the meantime, so there might be a another eBook later. Any feedback please to epldfpga@aol.com. Your own programs or applications you can forward to us, including please the ok to publish it on our website or in another eBook; if there is time we will keep a MyCo area on our website, have a look. Some more information you will find on our website www.exemark.com. Enjoy.

Specification:

Microcontroller: Holtek HT46F47
Clock frequency: about 2 MHz
Internal EEPROM: 128 Bytes
Power supply voltage: 2.2 V to 5.5 V
Current consumption: about 1 mA at 4.5 V
4 output pins: support up to 10 mA
1 PWM output: supports up to 10 mA
4 input lines: internal resistor sets to 1
2 analog inputs: 0 V ... Vcc
2 switch inputs: internal resistors set it to 1

Components in this learning package:

1 PCB
1 Holtek HT46F47 pre-programmed with TPS firmware
1 IC socket
3 Push buttons
4 LEDs 3 mm, red - short wire cathode (into square PCB hole)
1 LED 3 mm, green - short wire cathode (into square PCB hole)
1 LDR - Light Dependant Resistor
1 Piezo transducer
3 Capacitors 100 nF
1 Polarised capacitor 47 uF
5 Resistors 2.2 kOhm - red red red plus other colors
1 Resistor 10 kOhm - brown black orange plus other colors
1 Resistor 27 kOhm - red violet orange plus other colors
2 Resistors 100 kOhm - black brown yellow plus other colors
1 Wire 1m
1 Battery compartment with wires, for 3 AA batteries

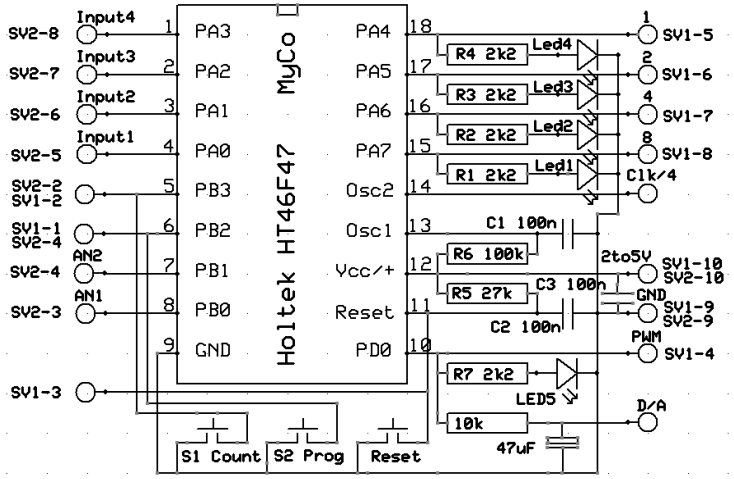


Figure 1.1: Circuit diagram including SV1 and SV2 connections

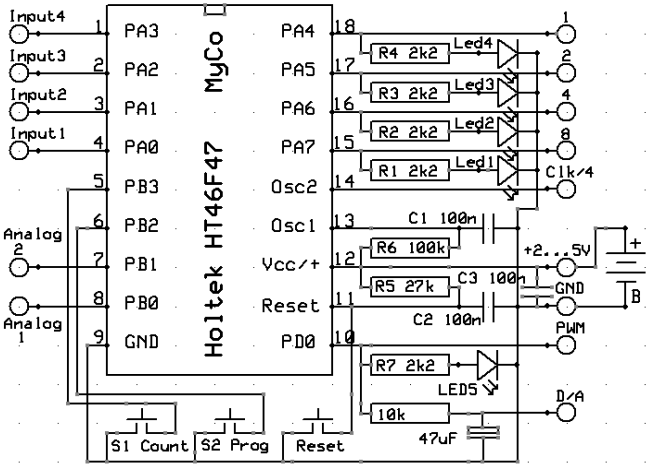


Figure 1.2: Circuit diagram of the board, reduced to main functions

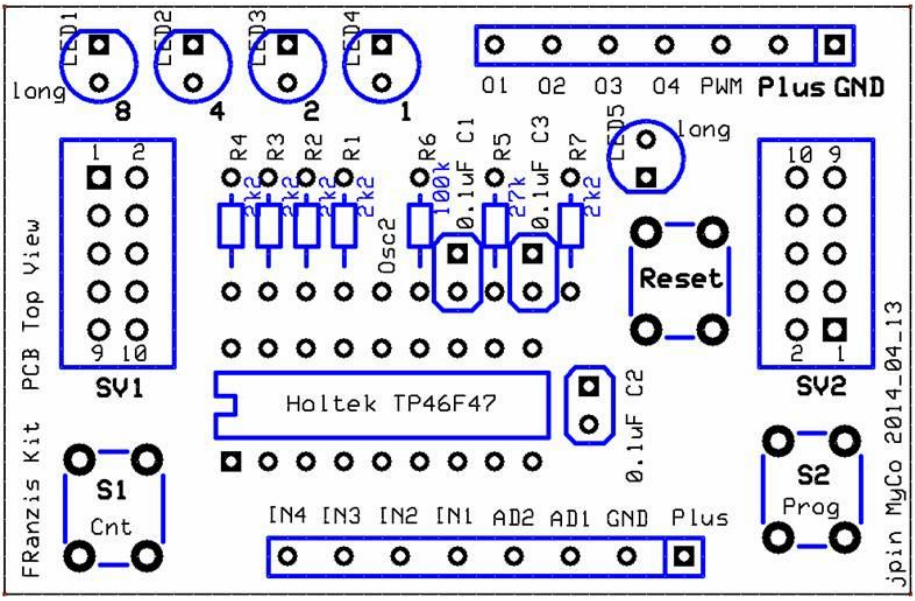


Figure 1.3: Component locations top view of the board, about 40 x 60 mm

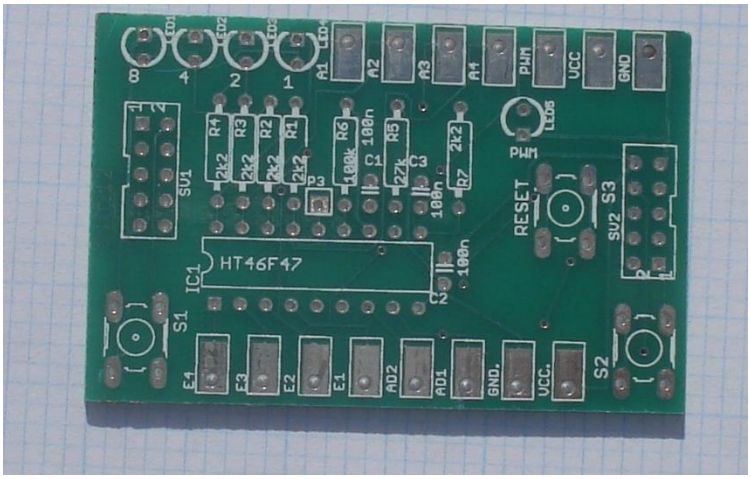


Figure 1.4t: The blank PCB, with the 4 Output LEDs 8,4,2,1 top left

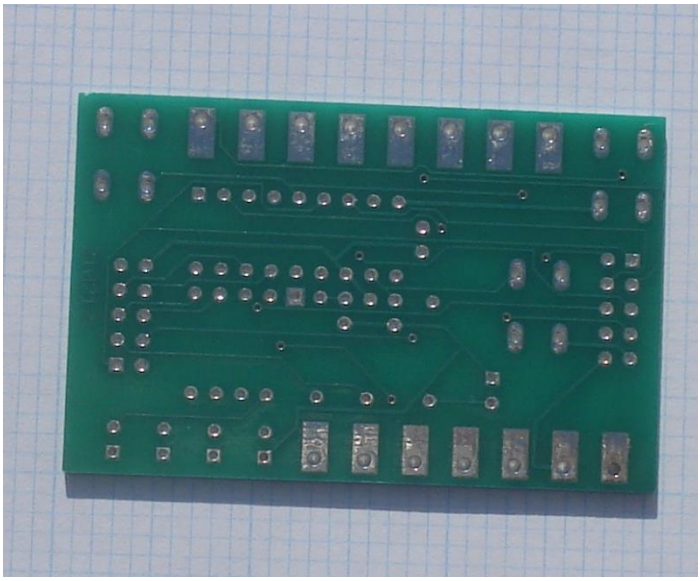


Figure 1.4b: Bottom side of the blank PCB

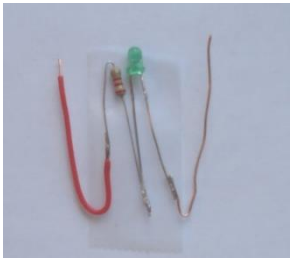


Figure 1.4l: Additional LED tester using the PWM resistor and LED

.

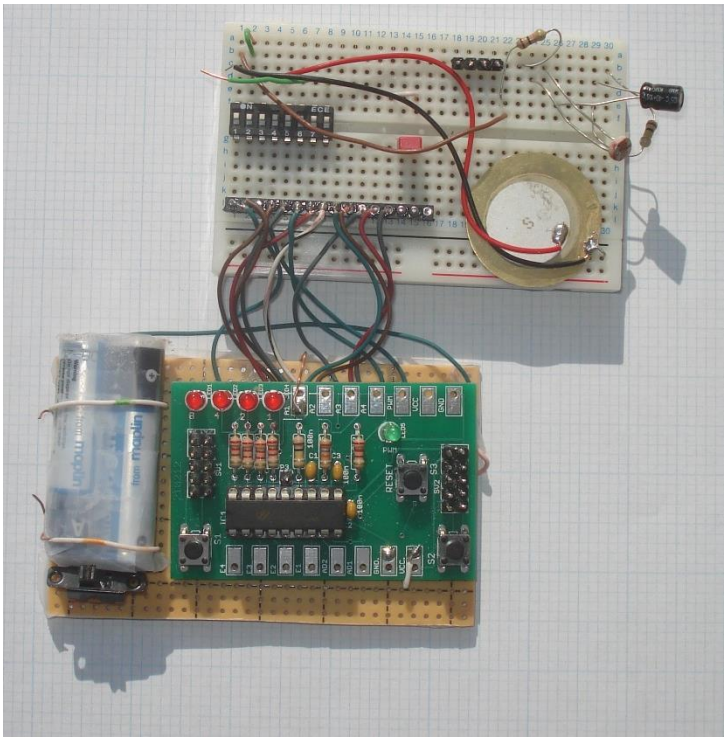


Figure 1.5: Standard setup with push-button switches and my added extension board

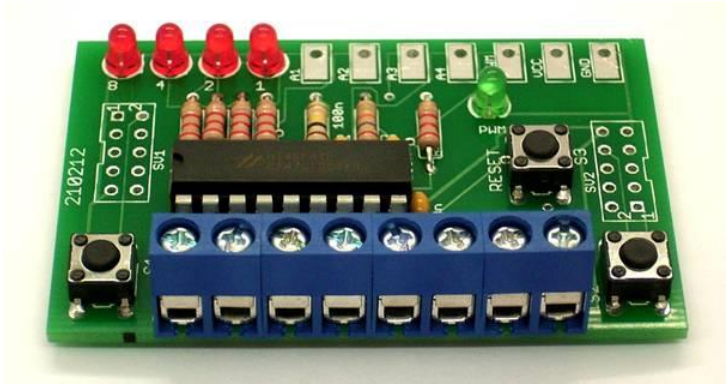


Figure 1.6: The use of screw terminals on top side

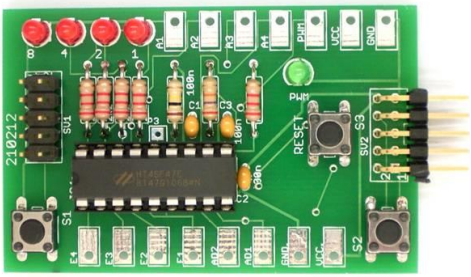


Figure 1.7: Use of the two pin header connectors

S2	1	2	S1
Reset	3	4	PWM
A1	5	6	A2
A3	7	8	A4
GND	9	10	VCC

Header connector SV1, as on the board

VCC	10	9	GND
E4	8	7	E3
E2	6	5	E1
AD2	4	3	AD1
S1	2	1	S2

Header connector SV2, (as on the board, turned 180º)

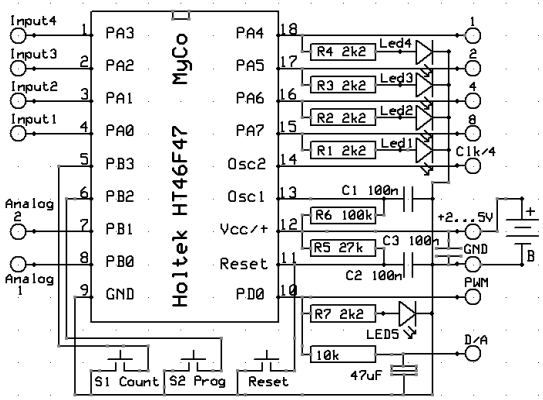


Figure 2.1: Four LEDs on the outputs

2.2 - Two flashing LEDs

Address	Instruction	Data	Comment
20	1	1	LED 1 on, out 0001
21	2	8	Wait for 500 ms
22	1	8	LED 8 on, out 1000
23	2	8	Wait for 500 ms
24	3	4	Jump to - 4

Listing 2.1: Switch on one LED and then another one

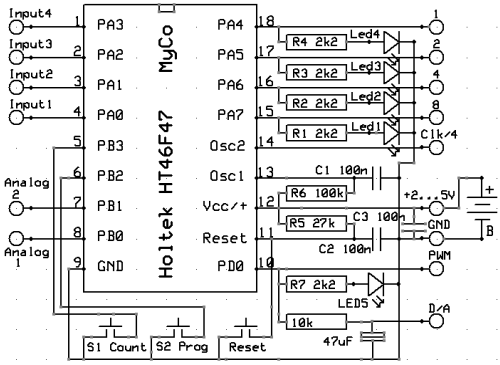


Figure 2.2: No extra wiring for this program listing 2.1

2.3 - Counter combined with PWM

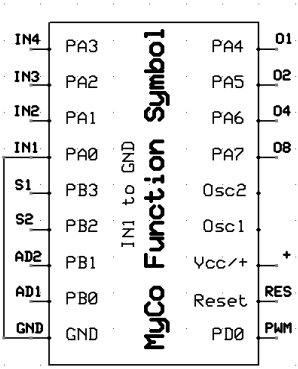


Figure 2.3: Counter and PWM function shown via the LED

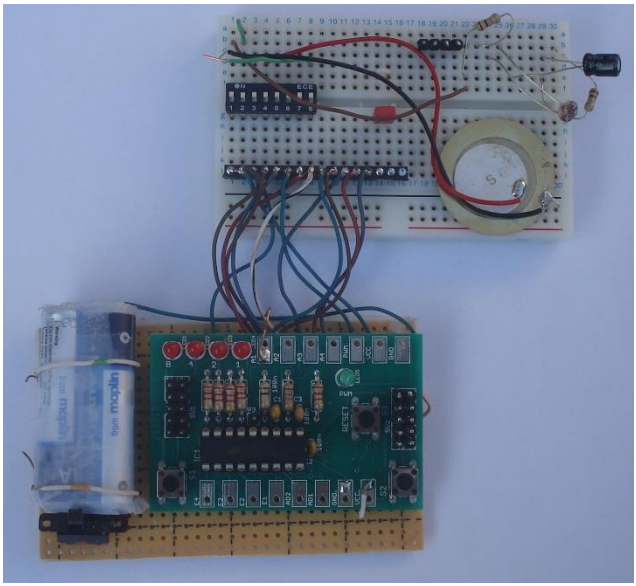
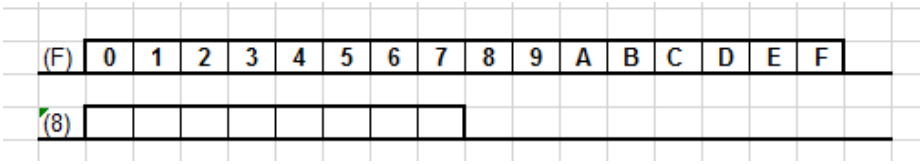


Figure 2.4: The binary counter



Address	Instruction	Data	Comment
25	7	1	A <= A + 1
26	5	4	Port <= A
27	5	9	PWM <= A
28	2	6	Wait 100 ms
29	3	4	Jump to - 4

Listing 2.2: Binary counter with LED and PWM output

„8“	„4“	„2“	„1“	Decimal	Hexadecimal
0	0	0	0	0	0
0	0	0	1	1	1
0	0	1	0	2	2
0	0	1	1	3	3
0	1	0	0	4	4
0	1	0	1	5	5
0	1	1	0	6	6
0	1	1	1	7	7
1	0	0	0	8	8
1	0	0	1	9	9
1	0	1	0	10	A
1	0	1	1	11	B
1	1	0	0	12	C
1	1	0	1	13	D
1	1	1	0	14	E
1	1	1	1	15	F

4 bit in binary, decimal and hexadecimal

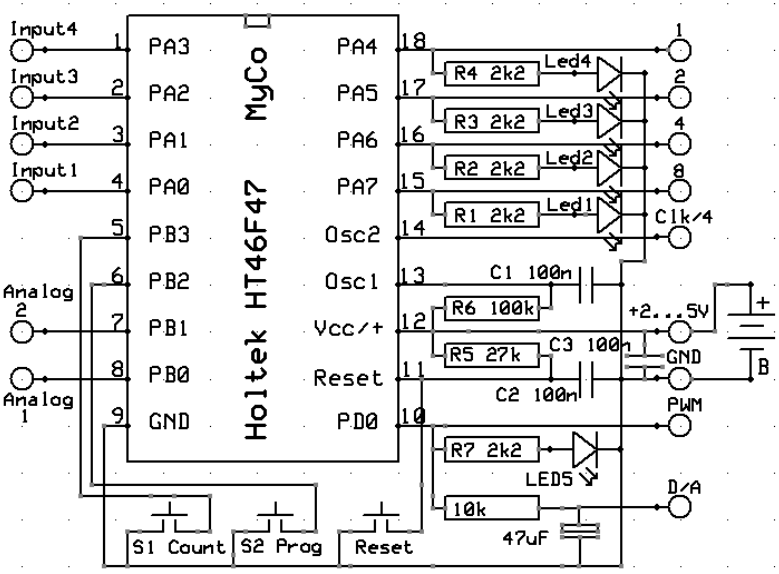


Figure 2.5: Low-pass filter on the PWM output

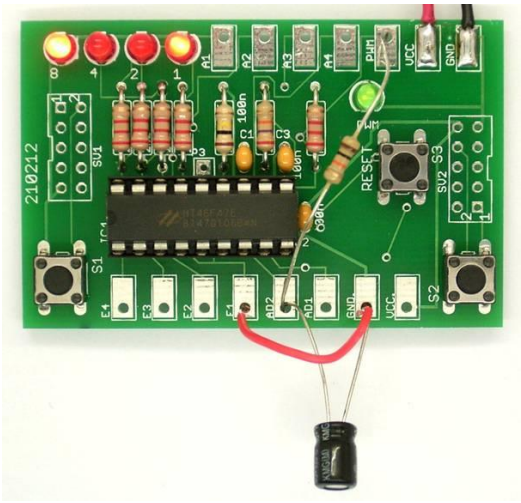


Figure 2.6: Smoothened PWM output voltage as analog output

2.4 - The Analog-to-Digital Converter

A table of the input voltage and the expected LED display if supply voltage is 4.0V:

```
Step 0    1    2    3    4    5    6    7
Volt 0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75

Step 8    9    A    B    C    D    E    F
Volt 2.00 2.25 2.50 2.75 3.00 3.25 3.50 3.75
```

Measure the actual voltage at the input with a high impedance input voltmeter to compare. Other resistors than 10k would give a different response to a given range of light.

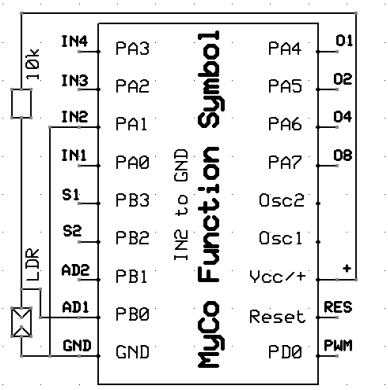


Figure 2.7: Connection of the light sensor

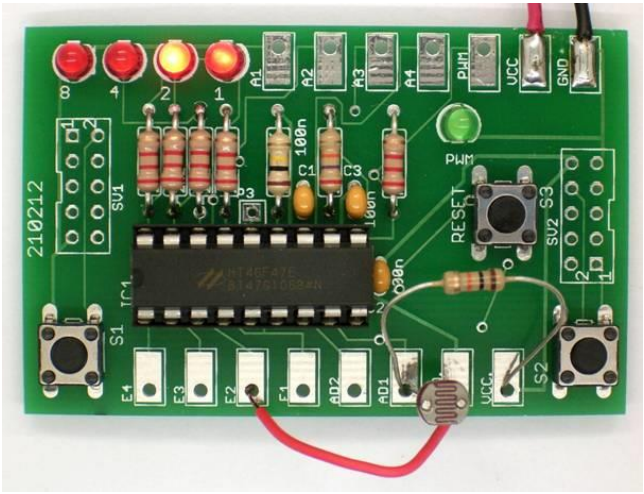


Figure 2.8: The LDR connected to input of AD1

Address	Instruction	Data	Comment
2A	6	9	A <= AD1
2B	5	4	Port <= A
2C	5	9	PWM <= A
2D	2	6	Wait 100 ms
2E	3	4	Jump to - 4

Listing 2.3: AD converter and PWM output

2.5 - Random Number Generator

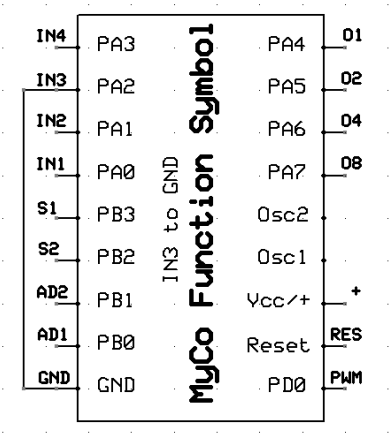


Figure 2.9: Start of the random number generator

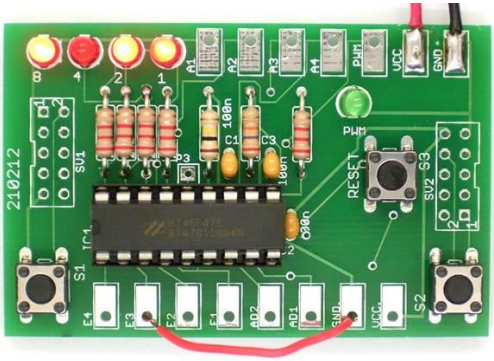
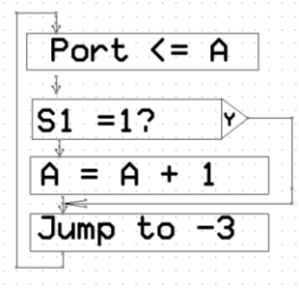


Figure 2.10: Wire bridge between E3 and GND

Address	Instruction	Data	Comment
30	5	4	Port <= A
31	C	E	S1 = 1?
32	7	1	A<= A + 1
33	3	3	Jump to - 3

Listing 2.4: Random Number Generator



Program flow shown in a different way

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

2.6 - Pulse Length Measurement

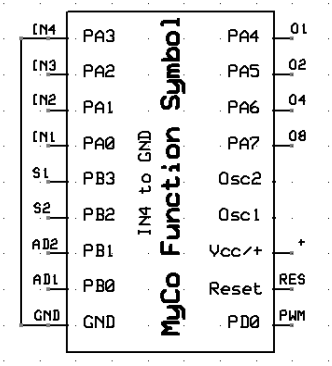


Figure 2.11: E4 to GND

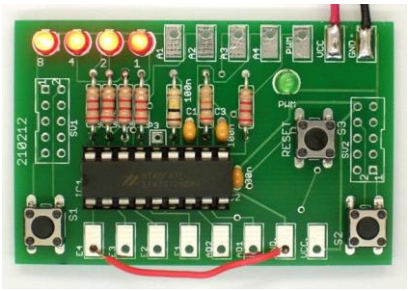


Figure 2.12: Pulse length measurement using Reset and S1

Address	Instruction	Data	Comment
34	2	2	Wait for 5 ms
35	C	C	S1 = 0?
36	3	2	Jump to - 2
37	4	0	A <= 0
38	2	2	Wait for 5 ms
39	7	1	A <= A + 1
3A	5	4	Port <= A
3B	C	E	S1 = 1?
3C	3	4	Jump to - 4
3D	3	9	Jump to - 9

Listing 2.5: Time measurement

3 - The Programming Mode

3.1 - First - Reading out Programs

6 4 5 1 4 E 80 C 3 9 8 …

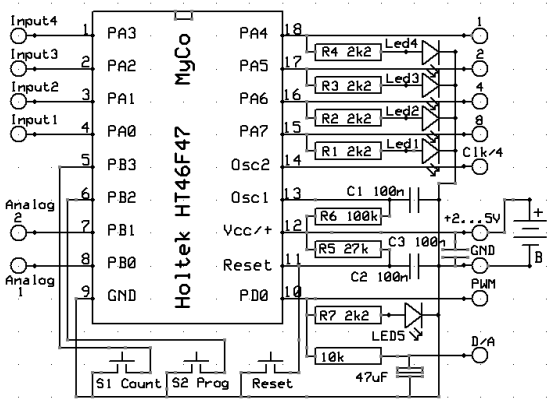


Figure 3.1: S1 and S2 and Reset for the programming mode

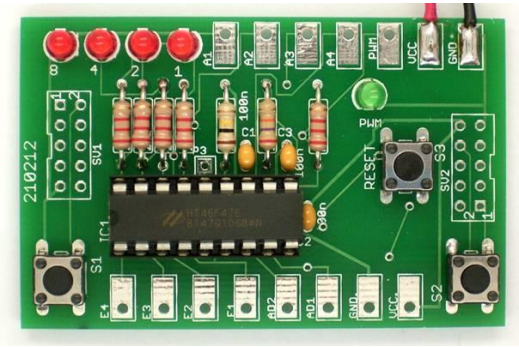


Figure 3.2: The three buttons, and the 4 LEDs top left are used

Address	Instruction	Data	Comment
00	6	4	A <= Din
01	5	1	B <= A
02	4	E	A <= 14
03	8	0	AddrHi <= 0
04	C	3	A = B?

Listing 3.1: Program code in pre-programmed controller

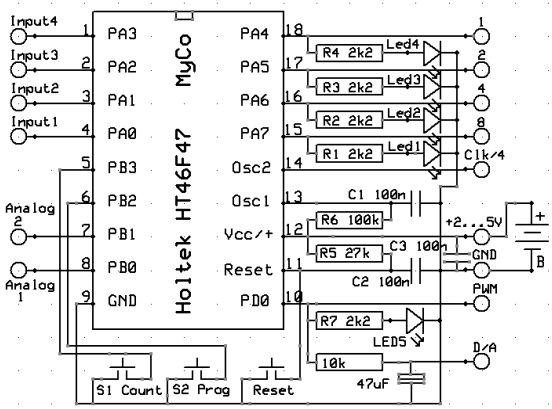
Just to put it into perspective:

128 byte EEPROM in MyCo as the program area for you
128 Kilobyte would be 1000 times this
128 Megabyte is 1000 x 1000 times more.
128 Gigabyte is 1000 x 1000 x 1000 times what we use here. (USB sticks have now mostly 4 to 32GB)

Morse Code

A	- ----	J	- ---- ---- ----	S	- - -	1	- ---- ---- ---- ----
B	---- - - -	K	---- - ----	T	----	2	- - ---- ---- ----
C	---- - ---- -	L	- ---- - -	U	- - ----	3	- - - ---- ----
D	---- - -	M	---- ----	V	- - - ----	4	- - - - ----
E	-	N	---- -	W	- ---- ----	5	- - - - -
F	- - ---- -	O	---- ---- ----	X	---- - - ----	6	---- - - - -
G	---- ---- -	P	- ---- ---- -	Y	---- - ---- ----	7	---- ---- - - -
H	- - - -	Q	---- ---- - ----	Z	---- ---- - -	8	---- ---- ---- - -
I	- -	R	- ---- -			9	---- ---- ---- ---- -
						0	---- ---- ---- ---- ----

3.2 - Programming new Functions



Address	Instruction	Data	Comment
00	1	7	A1...4 <= 0111
01	3	0	Jump to - 0

Listing 3.2: Turn on 3 LEDs

Address	Instruction	Data	Comment
00	6	4	A <= Din
01	5	4	Dout <= A
02	3	2	Jump to -2

Listing 3.2a: Status of IN1, IN2, IN3, IN4 the 4 LEDs

S2 + Reset	to put into Programming Mode
2 x S1	to reset counter and increase to 1
S2	program in this 1 and switch to the second nibble
8 x S1	reset counter to 0, then 7x to get to 7
S2	program this 7 in, increment to address 01
4 x S1	reset counter for instruction, 3x to get to 3
S2	program 3 in and change to data
1 x S1	only 1x S1 to reset internal counter
S2	program 0 in

3.3 - Back to Factory Status

Address	Instruction	Data	Comment
00	F	F	-
01	F	F	-

Listing 3.3: Return controller to the pre-programmed state

4 - MyCo Instructions

4.1 - The Basic Commands MyCo can execute

and the data flow. Input and output we know already, each 4 bit wide.

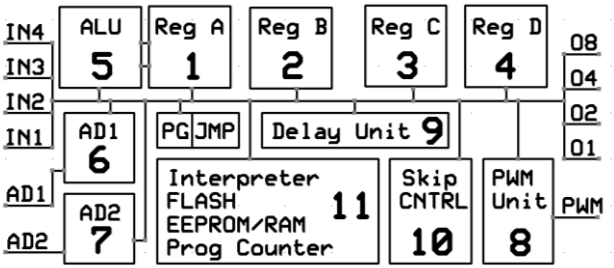


Figure 4.1a: Programming model as block

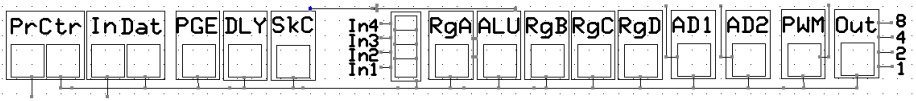


Figure 4.1b: Programming model in string form for program execution on paper

The different blocks in this string:

PrCtr Program counter needs 2 nibbles
In Dat Instruction and Data nibbles
PGE Page register
DLY Delay time
SkC Skip Control bit, if Yes then skip

In_n 4 input lines
RgA Register A
ALU Arithmetic and Logic Unit
RgB Register B
RgC Register C
RgD Register D
AD1 Analog-to-digital converter 1
AD2 Analog-to-digital converter 2
PWM Pulse Width Modulation output

OUT Output register
1248 4 bit output and LEDs

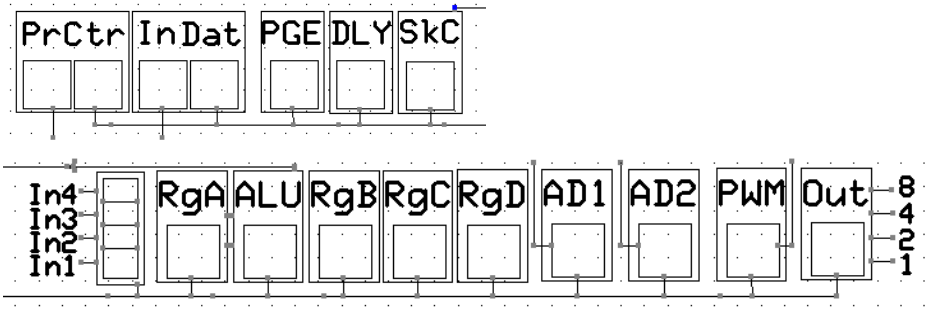


Figure 4.1c: Programming model as 2 blocks for program execution on paper

1 is easy to remember One, the first letter gives you the O for output
2 is as simple, because wait takes TOO long
3 is a bit more difficult, but for now you THRow the ball back

Address	Instruction	Data	Comment
00	1	1	A1...4 <= 0001
01	2	8	Wait for 500 ms
02	1	8	A1...4 <= 1000
03	2	8	Wait for 500 ms
04	3	4	Jump to - 4

Listing 4.1: Blink program, hex 11 28 18 28 34

Address	Instruction	Data	Comment
00	1	1	LEDs to 0001

01	2	8	Wait for 500 ms
02	1	2	LEDs to 0010
03	2	8	Wait for 500 ms
04	1	4	LEDs to 0100
05	2	8	Wait for 500 ms
06	1	8	LEDs to 1000
07	2	8	Wait for 500 ms
08	3	8	Jump to - 8

11 28 12 28 14 28 18 28 38

Listing 4.2: Running Light 1

Address	Instruction	Data	Comment
00	1	1	LEDs 0001
01	2	8	Wait for 500 ms
02	1	2	LEDs 0010
03	2	8	Wait for 500 ms
04	1	4	LEDs 0100
05	2	8	Wait for 500 ms
06	1	8	LEDs 1000
07	2	8	Wait for 500 ms
08	1	4	LEDs 0100
09	2	8	Wait for 500 ms
0A	1	2	LEDs 0010
0B	2	8	Wait for 500 ms
0C	3	C	Jump to - 12

11 28 12 28 14 28 18 28 14 28 12 28 3C

Listing 4.3: Running Light 2, right to left and back

Address	Instruction	Data	Comment
00	1	F	LEDs to 1111
01	2	F	Wait for 1 min
02	1	0	LEDs to 0000
03	3	0	End

1F 2F 10 30

Listing 4.4: Timer for one minute

4.2 - Calculations using Variables

Address	Instruction	Data	Comment
00	4	0	A <= 0
01	7	1	A <= A + 1
02	5	4	Port <= A
03	5	9	PWM <= A
04	2	6	Wait for 100 ms
05	3	4	Jump to - 4

Program code: 40 71 54 59 26 34

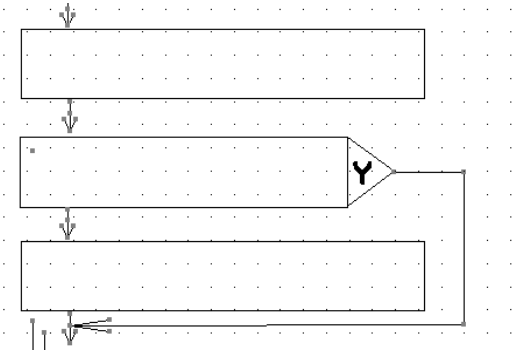
Listing 4.5: Increment by one, show on LEDs and PWM

Address	Instruction	Data	Comment
00	6	9	A <= AD1
01	5	4	Port <= A
02	7	A	A <= Not A
03	5	9	PWM <= A
04	2	6	Wait 100 ms
05	3	5	Jump to - 5

Program code: 69 54 7A 59 26 35

Listing 4.6: Inverting the contents of A

4.3 - Jumps and Skips



Flowchart 1: Do a test, SKIP over if Yes, else next instruction

Address	Instruction	Data	Comment
30	5	4	Port <= A
31	C	E	S1 = 1?
32	7	1	A<= A + 1
33	3	3	Jump to - 3

Listing 2.4: Random Number Generator

Address	Instruction	Data	Comment
00	C	C	S1 = 0?
01	3	1	Jump to - 1
02	4	0	A <= 0
03	7	1	A <= A + 1
04	5	4	Port <= A
05	C	E	S1 = 1?
06	3	3	Jump to - 3
07	3	7	Jump to - 7

CC 31 40 71 54 CE 33 37

Listing 4.7: S1 button response with higher resolution

Address	Instruction	Data	Comment
00	8	3	Set to page 3
01	9	4	Jump to Address = x4

83 94

Listing 4.8: Absolute Jump to pre-programmed timer program

4.4 - The Instruction Table

	1	2	3	4	5	6	7	8	9	A	B	C	D	E
	to Port	Wait	Jump back	A<= ...	... <=A	A<= ...	A<= ...	Set Page	Jump (Page)	C*	D*	Skip if ...	Call	Ret
0	0	1 ms	0	0				0	0	0	0		0	
1	1	2 ms	1	1	B<=A	A<= B	A<=A+1	1	1	1	1	A>B	1	
2	2	5 ms	2	2	C<=A	A<=C	A<=A–1	2	2	2	2	A<B	2	
3	3	10	3	3	D<=A	A<=D	A<=A+B	3	3	3	3	A=B	3	
4	4	20	4	4	Dout<=A	A<=Din	A<=A–B	4	4	4	4	Din.0=1	4	
5	5	50	5	5	A.0	A<=Din.0	A<= A*B	5	5	5	5	Din.1=1	5	
6	6	100	6	6	A.0	A<=Din.1	A<=A/B	6	6	6	6	Din.2=1	6	
7	7	200	7	7	A.0	A<=Din.2	B	7	7	7	7	Din.3=1	7	
8	8	500	8	8	A.0	A<=Din.3	A<=A Or B		8	8	8	Din.0=0	8	
9	9	1 s	9	9	PWM<=A	A<=AD1	A<= A Xor		9	9	9	Din.1=0	9	
A	10	2 s	10	10		A<=AD2	A<=Not A		A	A	A	Din.2=0	A	
B	11	5 s	11	11					B	B	B	Din.3=0	B	
C	12	10 s	12	12					C	C	C	S1=0	C	
D	13	20 s	13	13					D	D	D	S2=0	D	
E	14	30 s	14	14					E	E	E	S1=1	E	
F	15	60 s	15	15					F	F	F	S2=1	F	

5 - Program Structures and Sample Programs

5.1 - Counting Loops

Address	Instruction	Data	Comment
00	4	5	A <= 5
01	5	2	C <= A
02	1	5	Port <= 0101
03	2	8	Wait 500 ms
04	1	A	Port <= 1010
05	2	8	Wait 500 ms
06	8	0	Set to Page 0
07	A	2	C times 02
08	3	0	End

45 52 15 28 1A 28 80 A2 30

Listing 5.1: A timing loop

Address	Instruction	Data	Comment
00	4	5	A <= 5
01	5	2	C <= A
02	8	0	AddrHi <=0
03	A	5	C-times 05, skip if not 0
04	3	0	End
05	1	5	Port <= 0101
06	2	8	Wait for 500 ms
07	1	A	Port <= 1010
08	2	8	Wait for 500 ms
09	3	6	Jump to - 6

45 52 80 A5 30 15 28 1A 28 36

Listing 5.2: Five times flashing

5.2 - Compare Instructions

Address	Instruction	Data	Comment
00	4	5	A <= 5
01	5	1	B <= A
02	8	0	AddrHi = 0
03	6	9	A <= AD1
04	C	1	Skip if A>B
05	9	8	Page Addr 08
06	1	F	LEDs 1111
07	3	4	Jump to Addr 03
08	1	0	Set LEDs to 0000
09	3	6	Addr 03, 6 back

45 51 80 69 C1 98 1F 34 10 36

Listing 5.3: Simple twilight switch

5.3 - Single Bit Processing

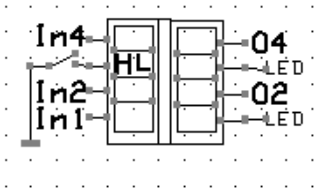


Figure 5.1: Testing of E3/In3/Din.2

Address	Instruction	Data	Comment
00	6	7	A <= Din.2 (E3)
01	5	4	Port <= A
02	2	1	Wait 2 ms
03	3	3	Jump to - 3

67 54 21 33

Listing 5.4: Single bit testing and display

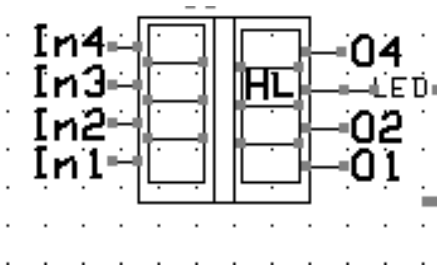


Figure 5.2: Controlling output A3/O3

Address	Instruction	Data	Comment
00	7	1	A <= A + 1
01	5	7	Port.2 <= A.0
02	2	8	Wait for 500 ms
03	3	3	Jump to - 3

71 57 28 33

Listing 5.5: A blinking LED on A3/O3/4

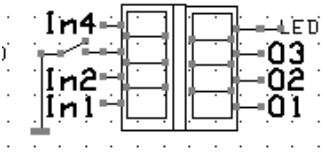


Figure 5.3: Inverted output

Address	Instruction	Data	Comment
00	6	7	A <= Din.2
01	7	A	A = NOT A
02	5	8	Port.3 <= A.0
03	3	3	Jump to - 3

67 7A 58 33

Listing 5.6: Invert a single bit and copy to O4/Port.3

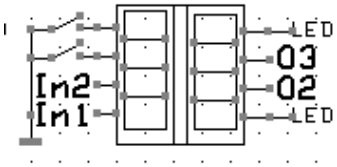


Figure 5.4: Change over switch with two outputs

Address	Instruction	Data	Comment
00	C	6	Skip if Din.2=1
01	1	1	Port <= 1 (0001)
02	C	7	Skip if Din.3=1
03	1	8	Port <= 8 (1000)
04	3	4	Go to Addr = 0

C6 11 C7 18 34

Listing 5.7: An RS flip-flop (Reset and Set)

5.4 - Basic Logic Functions

AND	X	Y	Result
	0	0	0
	0	1	0
	1	0	0
	1	1	1

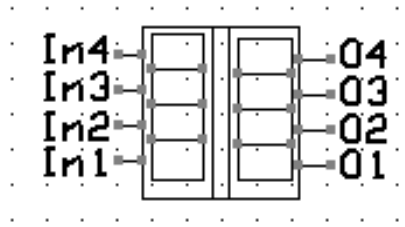


Figure 5.6: The four inputs E1…E4 (In1…In4) and 4 outputs A1…A4 (O1…O4), ANDing with 0011 Masking bit 0 and 1 in, masking bit 2 and 3 out

Address	Instruction	Data	Comment
00	6	4	A <= Din
01	5	1	B <= A
02	4	3	A <= 3 (0011)
03	7	7	A<= A AND B
04	5	4	Port <= A
05	3	5	Jump to - 5

64 51 43 77 54 35

Listing 5.8: Implementation of the AND function to mask bits

OR	X	Y	Result
	0	0	0
	0	1	1
	1	0	1
	1	1	1

XOR	X	Y	Result
	0	0	0
	0	1	1
	1	0	1
	1	1	0

INVERT		Y	Result
		0	1
		1	0

5.5 - Subroutines

Main program:

Address	Instruction	Data	Comment
00	8	0	AddrHi <=0
01	D	8	Call 08
02	5	4	Dout <= A
03	2	9	Wait for 1 s
04	D	8	Call 08
05	5	4	Dout <= A
06	2	8	Wait for 0,5 s
07	3	7	Jump to - 7

Subroutine:

Address	Instruction	Data	Comment
08	7	2	A <= A - 1
09	E	0	Return

80 D8 54 29 D8 54 28 37 72 E0 as main and sub follow each other in code space

Listing 5.9: Main program and subroutine calls

Address	Instruction	Data	Comment
00	4	0	A <= 0
01	5	4	Dout <= A
02	7	1	A <= A + 1
03	8	6	Set to Page 6
04	D	0	Call 60 to read S1
05	3	4	Jump to - 4

40 54 71 86 D0 34

Listing 5.10: Counter, controlled via S1

6 - Advanced Applications

6.1 - Twilight Switch

Address	Instruction	Data	Comment
00	1	0	Set LEDs to 0000
01	4	5	A <= 5
02	5	1	B <= A
03	6	9	A <= AD1
04	C	1	Skip if A>B
05	1	0	Set LEDs to 0000
06	4	9	A <= 9
07	5	1	B <= A
08	6	9	A <= AD1
09	C	2	Skip if A<B
0A	1	F	Set LEDs to 1111
0B	3	A	Jump to -10 =01

10 45 51 69 C1 10 49 51 69 C2 1F 3A

Listing 6.1: Twilight switch with hysteresis

6.2 - Two Point Controller

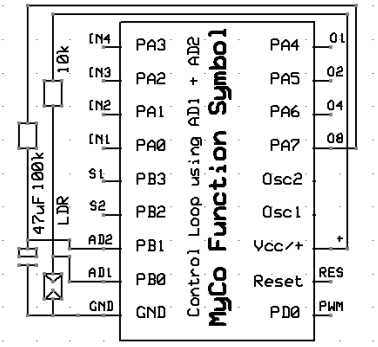


Figure 6.1: Control loop using AD2

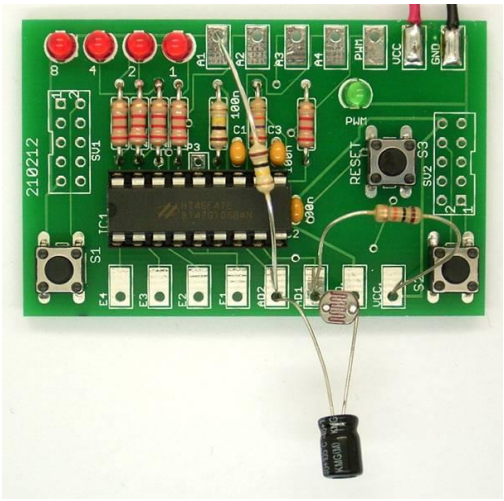


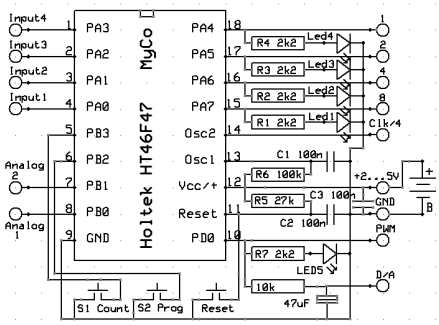
Figure 6.2: Voltage control via AD2

Address	Instruction	Data	Comment
00	6	9	A <= AD1
01	5	1	B <= A
02	8	0	AddrHi <= 0
03	6	A	A<=AD2
04	C	1	Skip if A>B
05	9	8	Addr 08
06	1	0	Output 0000
07	3	7	Jump to - 7
08	1	8	Output 1000
09	3	9	Jump to - 9

69 51 80 6A C1 98 10 37 18 39

Listing 6.2: The voltage follow loop

6.3 - LED Dimmer



Address	Instruction	Data	Comment
00	8	0	AddrHi <= 0
01	5	9	PWM <= A
02	2	7	Wait for 200 ms
03	5	2	C <= A
04	4	F	A <= 15
05	5	1	B <= A
06	6	2	A <= C
07	C	2	Skip if A<B
08	9	B	Jump to 0B
09	C	F	Skip if S2=1
0A	7	1	A <= A + 1
0B	5	2	C <= A
0C	4	0	A <= 0
0D	5	1	B <= A
0E	6	2	A <= C
0F	C	1	Skip if A>B
10	9	0	Jump to 00
11	C	E	Skip if S1 = 1
12	7	2	A <= A - 1
15	9	0	Jump to 00

80 59 27 52 4F 51 62 C2 9B CF 71 52 40 51 62 C1 90 CE 72 90

Listing 6.3: Brightness control

6.4 – Morse Code Program

Address	Instruction	Data	Comment
00	8	0	Set to page 0 >long
01	4	F	A <= 15
02	9	4	Jump to Addr 04
03	4	5	A <= 5 >short
04	5	3	D <= A >variable
05	1	8	Dout 8
06	1	0	Dout 0
07	2	1	Wait for 2 ms
08	1	8	Dout 8
09	1	0	Dout 0
0A	2	1	Wait for 2 ms
0B	1	8	Dout 8
0C	1	0	Dout 0
0D	2	0	Wait for 1 ms
0E	B	5	D * 5
0F	3	0	End here, stop

80 4F 94 45 53 18 10 21 18 10 21 18 10 20 B5 30

Listing 6.4: Test the sound output

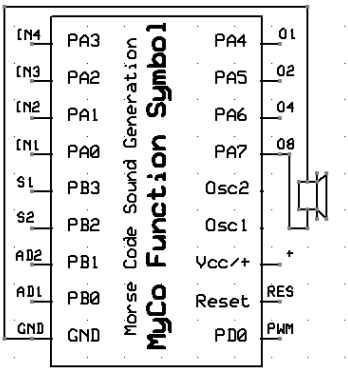


Figure 6.3 Connection of the piezo transducer

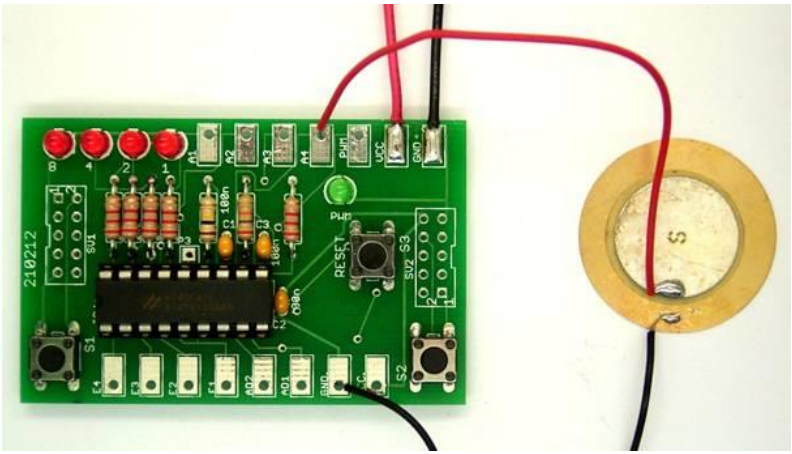


Figure 6.4: Sound output from A4

Morse Code : BK - long - short - short - short ---long - short - long

Address	Instruction	Data	Comment
00	8	5	AddrHi <=5
01	D	0	Call 50, long
02	2	6	100 ms
03	D	2	Call 52, short
04	2	6	100 ms
05	D	2	Call 52, short
06	2	6	100 ms
07	D	2	Call 52, short
08	2	6	100 ms
09	2	7	200 ms
0A	D	0	Call 50, long
0B	2	6	100 ms
0C	D	2	Call 52, short
0D	2	6	100 ms
0E	D	0	Call 50, long
0F	3	0	End, loop

85 D0 26 D2 26 D2 26 D2 26 27 D0 26 D2 26 D0 30

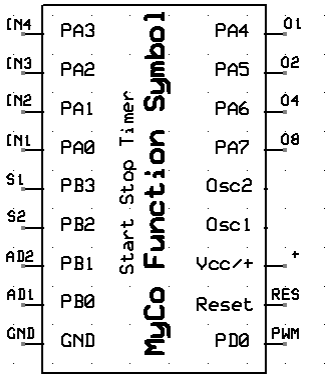
Listing 6.5: Morse code sound output

For more about Morse code, see Wikipedia:
http://en.wikipedia.org/wiki/Morse_code

Morse Code

A	- ▬	**J**	- ▬ ▬ ▬	**S**	- - -	**1**	- ▬ ▬ ▬ ▬
B	▬ - - -	**K**	▬ - ▬	**T**	▬	**2**	- - ▬ ▬ ▬
C	▬ - ▬ -	**L**	- ▬ - -	**U**	- - ▬	**3**	- - - ▬ ▬
D	▬ - -	**M**	▬ ▬	**V**	- - - ▬	**4**	- - - - ▬
E	-	**N**	▬ -	**W**	- ▬ ▬	**5**	- - - - -
F	- - ▬ -	**O**	▬ ▬ ▬	**X**	▬ - - ▬	**6**	▬ - - - -
G	▬ ▬ -	**P**	- ▬ ▬ -	**Y**	▬ - ▬ ▬	**7**	▬ ▬ - - -
H	- - - -	**Q**	▬ ▬ - ▬	**Z**	▬ ▬ - -	**8**	▬ ▬ ▬ - -
I	- -	**R**	- ▬ -			**9**	▬ ▬ ▬ ▬ -
						0	▬ ▬ ▬ ▬ ▬

6.5 - Start / Stop Timer



Address	Instruction	Data	Comment
00	8	6	Set to Page 6 “Timer Start/Stop”
01	D	0	Call “Wait for S1” on page 6
02	4	0	A <= 0
03	7	1	A <= A + 1
04	5	4	Port <= A

05	2	9	Wait for 1 s
06	C	D	S2 = 0?
07	3	4	Jump to - 4
08	D	8	Call “Wait S2”
09	4	0	A <= 0
0A	5	4	Port <= A
0B	3	B	Jump to - 11 (addr 00)

86 D0 40 71 54 29 CD 34 D8 40 54 3B

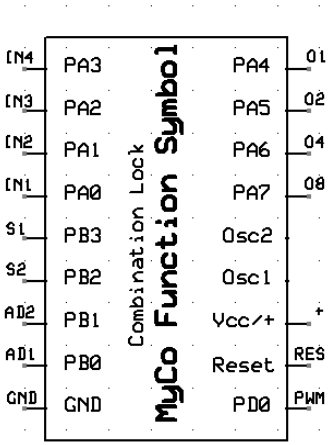
Listing 6.6: Stopwatch

Address	Instruction	Data	Comment
00	8	4	Set to Page 4
01	9	0	Jump to 40

84 90

Listing 6.7: Main program to start the stopwatch demo program

6.6 - Combination Lock



Address	Instruction	Data	Comment
00	C	C	Is S1 = 0?
01	3	1	Jump to - 1
02	4	0	A <= 0
03	5	4	Dout <= A
04	2	3	Wait for 10 ms
05	C	E	Is S1 = 1?
06	3	2	Jump to Addr 04
07	C	F	S2 = 1?
08	3	0	End
09	C	C	Is S1 = 0?
0A	3	3	Jump to Addr 07
0B	7	1	A <= A + 1
0C	2	3	Wait for 10 ms
0D	C	C	Is S1 = 1?
0E	3	1	Jump to Addr 0D
0F	3	C	Jump to Addr 03

CC 31 40 54 23 CE 32 CF 30 CC 33 71 23 CC 31 3C

Listing 6.8: Listing of the combination lock

Address	Instruction	Data	Comment
00	8	7	Set to Page 7
01	4	3	A <= 3
02	5	1	B <= A
03	D	0	Call 70
04	C	3	Skip if A=B
05	3	0	End - wrong
06	1	0	LEDs off
07	4	5	A <= 5
08	5	1	B <= A
09	D	0	Call 70
0A	C	3	Skip if A=B
0B	3	0	End - wrong

0C	1	0	LEDs off
0D	4	2	A <= 2
0E	5	1	B <= A
0F	D	0	Call 70
10	C	3	Skip if A=B
11	3	0	End - wrong
12	1	0	LEDs off
13	4	F	A <= 15
14	5	9	PWM <= A
15	3	0	End

87 43 51 D0 C3 30 10 45 51 D0 C3 30 10 42 51 D0 C3 30 10 4F 59 30

Listing 6.9: The combination lock

7 - The Inner Workings of MyCo

Having gone through the sample programs and having had a taste, you are probably now ready to understand most the inner workings better:

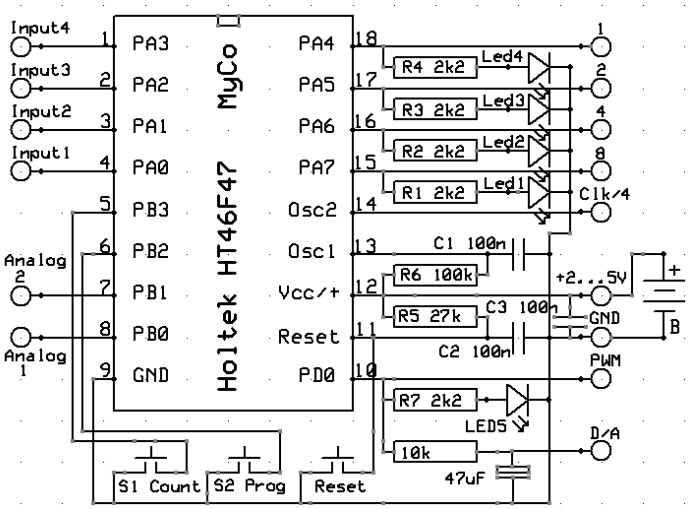


Figure 7.1: The circuit diagram

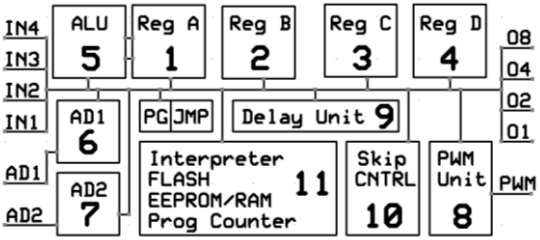


Figure 7.2: The inner functionality as block

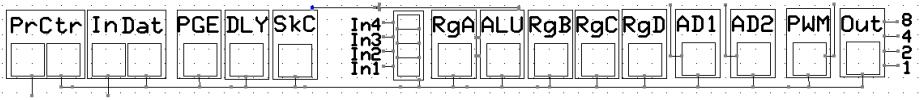


Figure 7.3a: The inner functionality all in a row

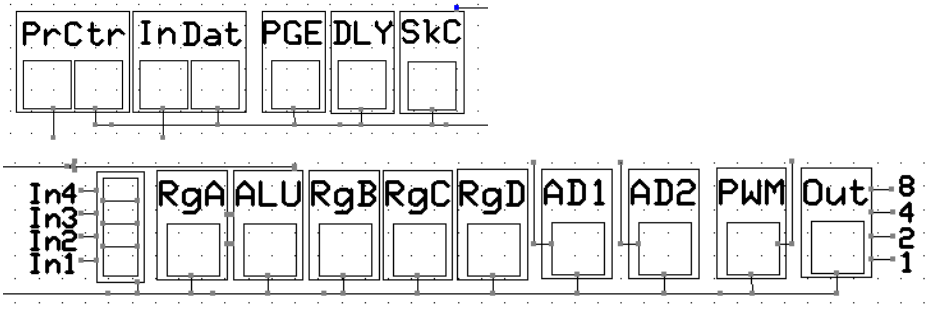


Figure 7.3b: The inner functionality broken into two rows

Any feedback to this eBook please send to epldfpga@aol.com.

We tried to eliminate as many issues, typos, … as possible.
But we know we are human, so errors are possible, but known ones can be corrected in the next edition, so please keep sending them in.

And from time to time have a look at our website www.exemark.com, we will try to add pages there with additional info.

8 - Appendix

8.1 - Listing of sample programs

Address	Instruction	Data	Comment
00	6	4	A <= Din
01	5	1	B <= A
02	4	E	A <= 1110
03	8	0	Page 0
04	C	3	A = B?
05	9	8	Jump to 08
06	8	2	Set page to 2
07	9	5	Jump to 05
08	4	D	A <= 1101
09	8	0	Set to Page 0
0A	C	3	A = B?
0B	9	E	Jump back E addresses
0C	8	2	Set to Page 2
0D	9	A	Jump to 2A, AD/PWM
0E	4	B	A <= 1011
0F	8	1	Set to Page 1

64 51 4E 80 C3 98 82 95 4D 80 C3 9E 82 9A 4B 81

Page 0 of the EEPROM, and starting from 0 after Reset: selection and start of example programs

---

Address	Instruction	Data	Comment
10	C	3	A = B?
11	9	4	Jump to 34
12	8	3	Set to Page 3
13	9	0	Jump to 30, “Random”
14	4	7	A <= 0111
15	8	1	Set to Page1

16	C	3	Is A =B?
17	9	A	Jump 1A
18	8	3	Set to Page 3
19	9	4	Jump to 34, “Stop S1”
1A	4	3	A <= 0011
1B	8	2	Set to 2
1C	C	3	Is A =B?
1D	9	0	Jump to 20 “LED blink”
1E	8	4	Set to Page 4
1F	9	0	Jump to 40 “Stop S1/S2”

C3 94 83 90 47 81 C3 9A 83 94 43 82 C3 90 84 90

Page 1: Select and run the example programs

Address	Instruction	Data	Comment
20	1	1	Output 0001 “2 LED Blink”
21	2	8	Wait for 500 ms
22	1	8	Output 1000
23	2	8	Wait for 500 ms
24	3	4	Jump 4 back 4
25	7	1	A <= A + 1 “Count”
26	5	4	Port <= A
27	5	9	PWM <= A
28	2	6	Wait for100 ms
29	3	4	Jump - 4
2A	6	9	A <= AD1 “AD/PWM”
2B	5	4	Port <= A
2C	5	9	PWM <= A
2D	2	6	Wait for 100 ms
2E	3	4	Jump by - 4
2F	F	F	-

11 28 18 28 34 71 54 59 26 34 69 54 59 26 34 FF

Page 2: Example programs: alternate flashing, counting, AD / PWM

Address	Instruction	Data	Comment
30	5	4	Port <= A "Random"
31	C	E	S1 <= 1?
32	7	1	A <= A + 1
33	3	3	Jump - 3
34	2	2	Wait for 5 ms "Stop on S1"
35	C	C	Is S1 = 0?
36	3	2	Jump by -2
37	4	0	A <= 0
38	2	2	Wait for 5 ms
39	7	1	A <= A + 1
3A	5	4	Port <= A
2B	C	E	S1 = 1?
3C	3	4	Jump by - 4
3D	3	9	Jump by - 9
3E	F	F	-
3F	F	F	-

54 CE 71 33 22 CC 32 40 22 71 54 CE 34 39 FF FF

Page 3: Example programs: random number, stopwatch S1

Address	Instruction	Data	Comment
40	8	6	Set to Page 6 "Start/Stop"
41	D	0	Call "Wait S1" at 60
42	4	0	A <= 0
43	7	1	A <= A + 1
44	5	4	Port <= A
45	2	3	Wait for10 ms

46	C	D	Is S2 = 0?
47	3	4	Jump by - 4
48	D	8	Call “Wait for S2”
49	4	0	A <= 0
4A	5	4	Port <= A
4B	3	B	Jump by - 11
4C	F	F	-
4D	F	F	-
4E	F	F	-
4F	F	F	-

86 D0 40 71 54 23 CD 34 D8 40 54 3B FF FF FF FF

Page 4: Example program stop watch start / stop

Address	Instruction	Data	Comment
50	4	F	A<=15 “Sound long”
51	9	3	Jump to Addr 03
52	4	5	A<=5 “Sound short”
53	5	3	D<=A “Sound variable”
54	1	9	A4 <= 1
55	1	1	A4 <= 0
56	2	1	2 ms delay
57	1	9	A4 <= 1
58	1	1	A4 <= 0
59	2	1	2 ms
5A	1	9	A4 <= 1
5B	1	1	A4 <= 0
5C	2	0	delay 1 ms
5D	B	4	D times 04
5E	1	0	Dout <=0
5F	E	0	Return

4F 93 45 53 19 11 21 19 11 21 19 11 20 B4 10 E0

Page 5: Subroutine sound output

Address	Instruction	Data	Comment
60	2	3	Wait 10 ms “Wait S1”
61	C	E	S1 = 1?
62	3	2	Jump to - 2
63	2	3	Wait for 10 ms
64	C	C	S1 = 0?
65	3	1	Jump to - 1
66	E	0	Return
67	F	F	-
68	2	3	Wait for 10 ms “Wait S2”
69	C	F	S2 = 1?
6A	3	2	Jump to - 2
6B	2	3	Wait for 10 ms
6C	C	D	S2 = 0?
6D	3	1	Jump to - 1
6E	E	0	Return
6F	F	F	-

23 CE 32 23 CC 31 E0 FF 23 CF 32 23 CD 31 E0 FF

Page 6: Subroutines waiting for S1 and for S2

Address	Instruction	Data	Comment
70	C	C	S1 = 0? “Switch Input“
71	3	1	Jump to - 1
72	4	0	A = 0
73	5	4	Port = A
74	2	3	Wait for 10 ms
75	C	E	S1 = 1?

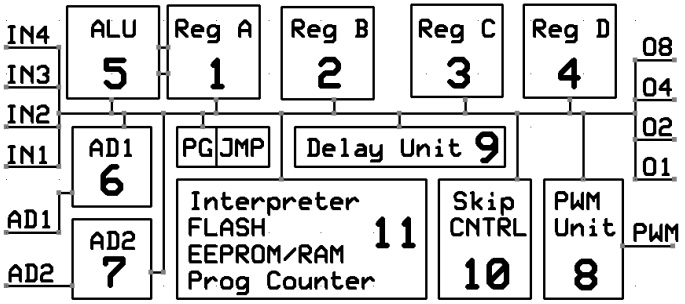
76	3	2	Jump to - 2
77	C	F	S2 = 1?
78	E	0	Return
79	C	C	S1 = 0?
7A	3	3	Jump to - 3
7B	7	1	A = A + 1
7C	2	3	Wait for 10 ms
7D	C	C	S1 = 1?
7E	3	1	Jump to - 1
7F	3	C	Jump to - 12

CC 31 40 54 23 CE 32 CF E0 CC 33 71 23 CC 31 3C

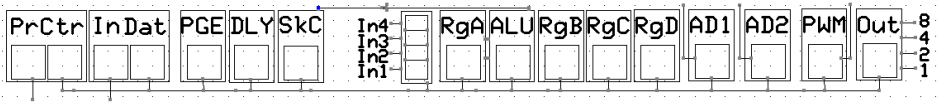
Page 7: Subroutine switch input

	1	2	3	4	5	6	7	8	9	A	B	C	D	E
	to Port	Wait	Jump back	A<= ...	... <=A	A<= ...	A<= ...	Set Page	Jump (Page)	C*	D*	Skip if ...	Call	Ret
0	0	1 ms	0	0				0	0	0	0		0	
1	1	2 ms	1	1	B<=A	A<= B	A<=A+1	1	1	1	1	A>B	1	
2	2	5 ms	2	2	C<=A	A<=C	A<=A–1	2	2	2	2	A<B	2	
3	3	10	3	3	D<=A	A<=D	A<=A+B	3	3	3	3	A=B	3	
4	4	20	4	4	Dout<=A	A<=Din	A<=A–B	4	4	4	4	Din.0=1	4	
5	5	50	5	5	A.0	A<=Din.0	A<= A*B	5	5	5	5	Din.1=1	5	
6	6	100	6	6	A.0	A<=Din.1	A<=A/B	6	6	6	6	Din.2=1	6	
7	7	200	7	7	A.0	A<=Din.2	B	7	7	7	7	Din.3=1	7	
8	8	500	8	8	A.0	A<=Din.3	A<=A Or B		8	8	8	Din.0=0	8	
9	9	1 s	9	9	PWM<=A	A<=AD1	A<= A Xor		9	9	9	Din.1=0	9	
A	10	2 s	10	10		A<=AD2	A<=Not A		A	A	A	Din.2=0	A	
B	11	5 s	11	11					B	B	B	Din.3=0	B	
C	12	10 s	12	12					C	C	C	S1=0	C	
D	13	20 s	13	13					D	D	D	S2=0	D	
E	14	30 s	14	14					E	E	E	S1=1	E	
F	15	60 s	15	15					F	F	F	S2=1	F	

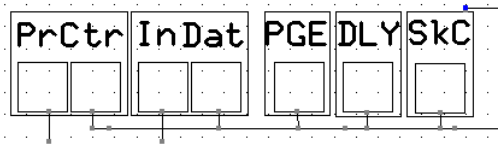
8.2 - Instruction Table

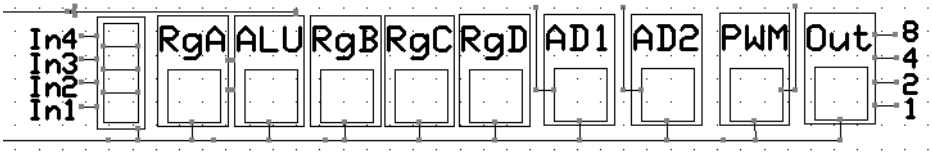


8.3 - Programming Model A

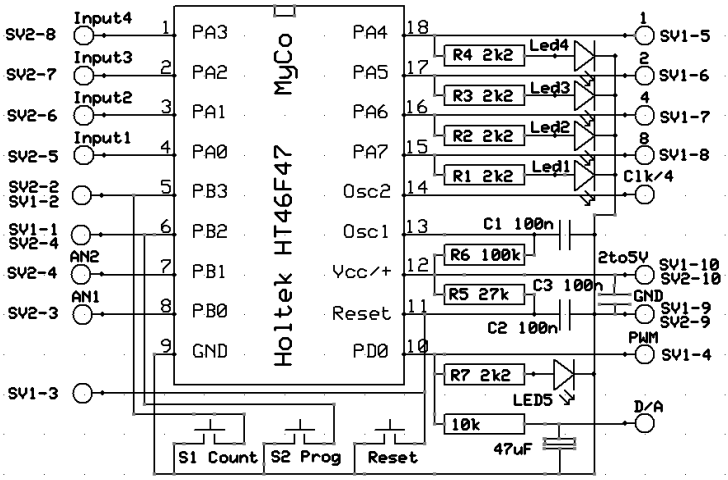


Programming Model B, all in one row

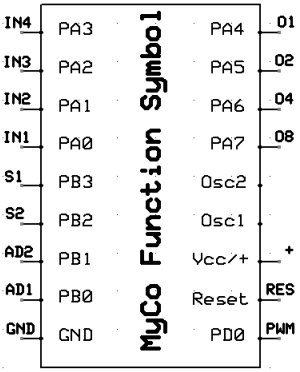




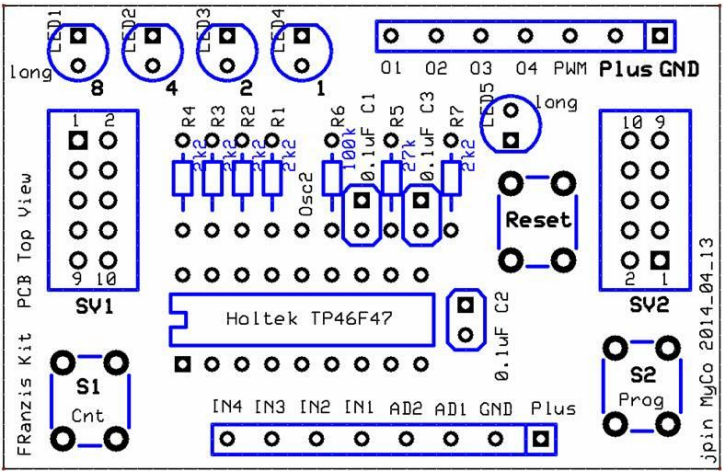
Programming model C, split up



8.4 - MyCo Circuit Diagram



8.5 - MyCo Function Symbol



8.6 - PCB top view with all components

S2	1	2	S1
Reset	3	4	PWM
A1	5	6	A2
A3	7	8	A4
GND	9	10	VCC

Header connector SV1
same way on the board

S2	1	2	S1
AD1	3	4	AD2
E1	5	6	E2
E3	7	8	E4
GND	9	10	VCC

Header connector SV2,
turned 180 degrees on the board
if populated, use angled version

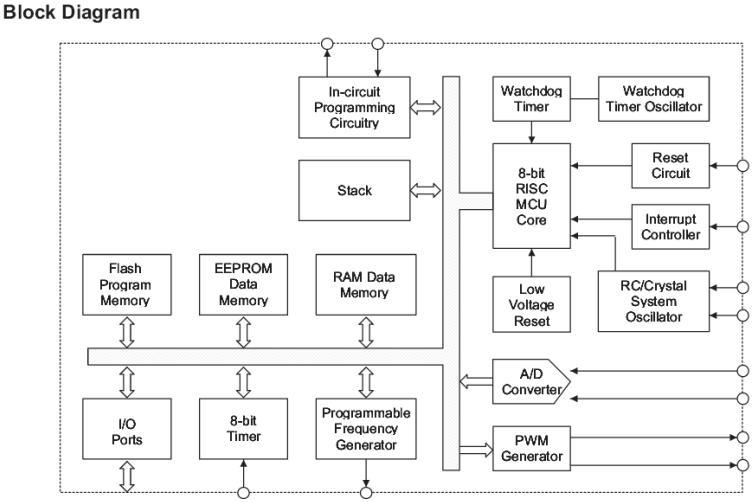
8.7 – Header Connectors

„8“	„4“	„2“	„1“	Decimal	Hexadecimal
0	0	0	0	0	0
0	0	0	1	1	1
0	0	1	0	2	2
0	0	1	1	3	3
0	1	0	0	4	4
0	1	0	1	5	5
0	1	1	0	6	6
0	1	1	1	7	7
1	0	0	0	8	8
1	0	0	1	9	9
1	0	1	0	10	A
1	0	1	1	11	B
1	1	0	0	12	C
1	1	0	1	13	D
1	1	1	0	14	E
1	1	1	1	15	F

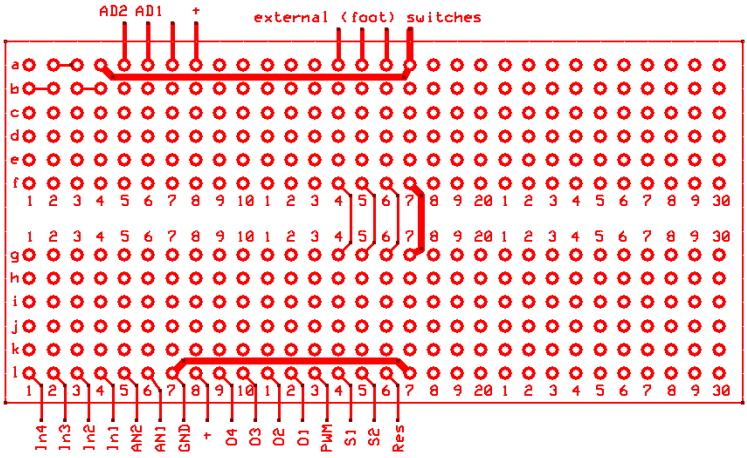
8.8 - Binary, Decimal and Hexadecimal Table

Link to the Holtek processor used here
http://www.holtek.com/english/docum/uc/46f4xe.htm

And the data sheet http://www.holtek.com/pdf/uc/46f4xev140.pdf



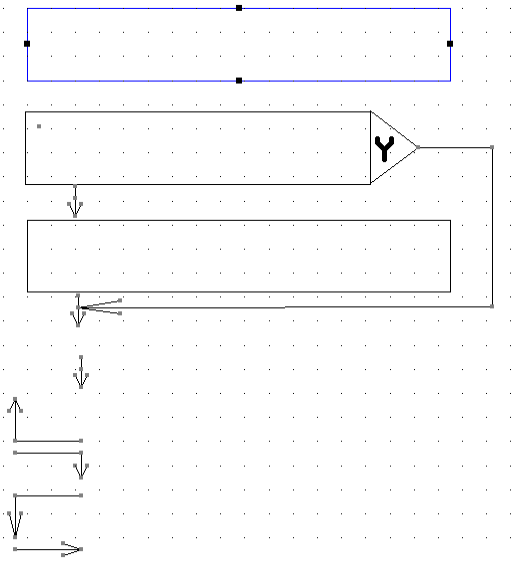
8.9 - Block diagram of the processor, source Holtek data sheet



8.10 - My Maplin breadboard, 16 connections of the Interface Connector

Colour	1st Band	2nd Band	3rd band	Multiplier	Tolerance
Black	0	0	0	1 Ohm	±1%
Brown	1	1	1	10 Ohm	±2%
Red	2	2	2	100 Ohm	
Orange	3	3	3	1kOhm	
Yellow	4	4	4	10kOhm	
Green	5	5	5	100kOhm	±0.5 %
Blue	6	6	6	1 MOhm	± 0.25%
Violet	7	7	7	10 MOhm	± 0.1%
Grey	8	8	8		±0.0.5%
White	9	9	9		
Gold				0.1 Ohm	±5%
Silver				0.01 Ohm	±10%

8.11 - Resistor Color Code (you might not have the Internet within reach)



8.12 - Toolbox to draw the flow diagrams:

Square box to describe what the Instruction does,
Square Box with Exit Y: if condition NO just continue, if YES skip over next Instruction
Arrow to next Instruction
Jump back
Come back in
Jump forward
Get back in line,
and next would be the small arrow down into the next block

8.13 - Interfacing MyCo to the PC to download programs

COM port (direct port access)

Included in version:
DMM-ProfiLab: Yes Digital-ProfiLab: Yes ProfiLab-Expert: Yes

The COM port may be used to control external hardware as well. But level shifting is needed.
The following pins are useable at the serial COM port:

4 digital inputs (CTS, DSR, RI, DCD) read in the outputs via transistor interface
3 digital outputs (DTR, RTS, TxD) control MyCo via 3 switches connected via level shifters.

The pin assignment of the COM port depends on the connector (9 pins or 25 pins):

Connector with 25 pins:

CTS	Pin 5

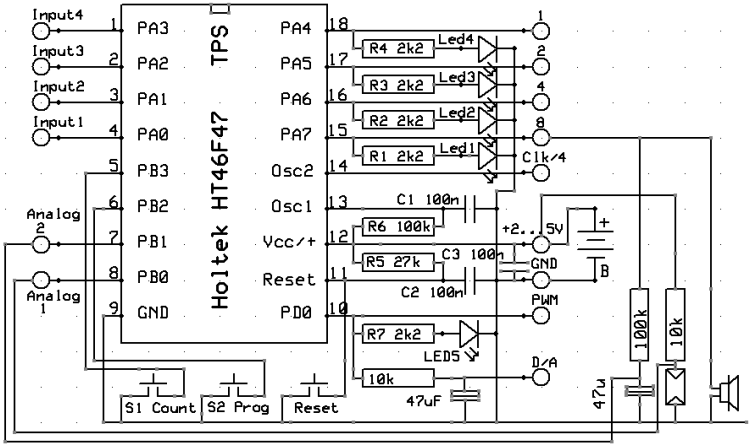
DSR	Pin 6
RI	Pin 22
DCD	Pin 8

DTR	Pin 20
RTS	Pin 4
TxD	Pin 2

Connector with 9 pins:

CTS	Pin 8
DSR	Pin 6
RI	Pin 9
DCD	Pin 1

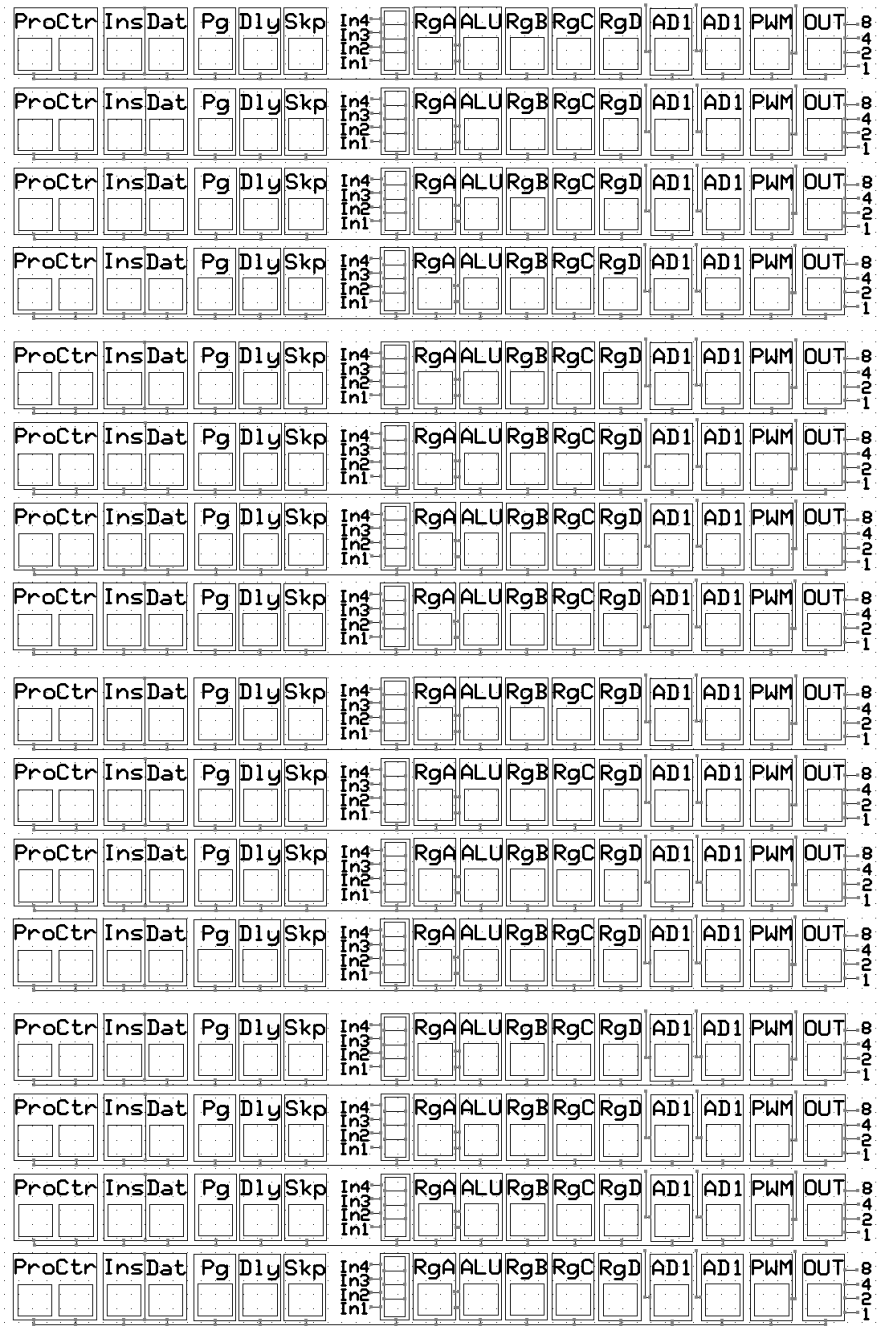
DTR	Pin 4
RTS	Pin 7
TxD	Pin 3

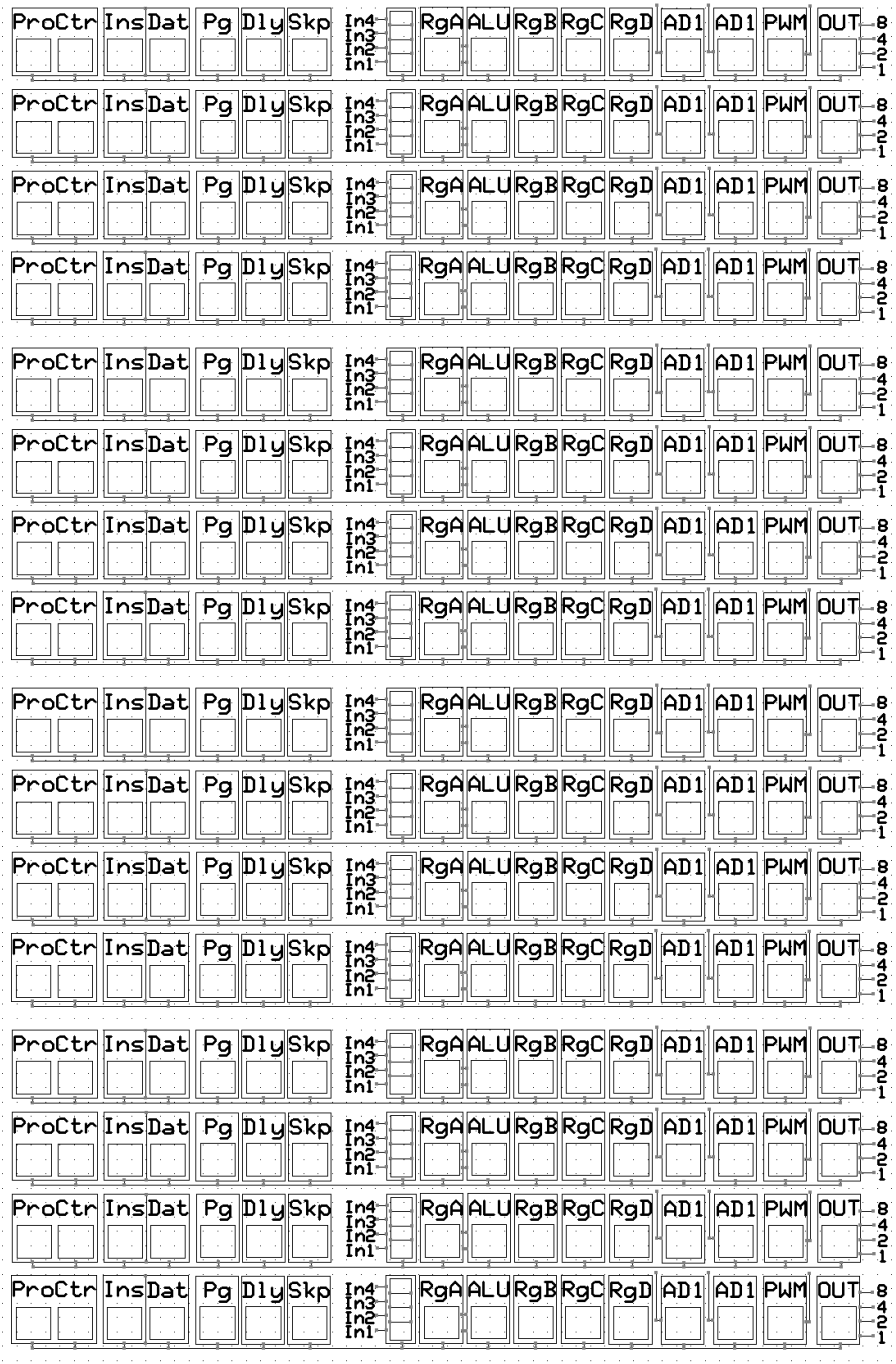


8.14 - Circuit diagram plus external parts

	1	2	3	4	5	6	7	8	9	A	B	C	D	E
	to Port	Wait	Jump back	A<= ...	... <=A	A<= ...	A<= ...	Set Page	Jump (Page)	C*	D*	Skip if ...	Call	Ret
0	0	1 ms	0	0				0	0	0	0		0	
1	1	2 ms	1	1	B<=A	A<= B	A<=A+1	1	1	1	1	A>B	1	
2	2	5 ms	2	2	C<=A	A<=C	A<=A–1	2	2	2	2	A<B	2	
3	3	10	3	3	D<=A	A<=D	A<=A+B	3	3	3	3	A=B	3	
4	4	20	4	4	Dout<=A	A<=Din	A<=A–B	4	4	4	4	Din.0=1	4	
5	5	50	5	5	A.0	A<=Din.0	A<= A*B	5	5	5	5	Din.1=1	5	
6	6	100	6	6	A.0	A<=Din.1	A<=A/B	6	6	6	6	Din.2=1	6	
7	7	200	7	7	A.0	A<=Din.2	B	7	7	7	7	Din.3=1	7	
8	8	500	8	8	A.0	A<=Din.3	A<=A Or B		8	8	8	Din.0=0	8	
9	9	1 s	9	9	PWM<=A	A<=AD1	A<= A Xor		9	9	9	Din.1=0	9	
A	10	2 s	10	10		A<=AD2	A<=Not A		A	A	A	Din.2=0	A	
B	11	5 s	11	11					B	B	B	Din.3=0	B	
C	12	10 s	12	12					C	C	C	S1=0	C	
D	13	20 s	13	13					D	D	D	S2=0	D	
E	14	30 s	14	14					E	E	E	S1=1	E	
F	15	60 s	15	15					F	F	F	S2=1	F	

8.14 - Instruction Table


ProCtr Ins Dat Pg Dly Skp In4 In3 In2 In1 RgA ALU RgB RgC RgD AD1 AD1 PWM OUT 8 4 2 1


ProCtr
Ins
Dat
Pg
Dly
Skp
In4
In3
In2
In1
RgA
ALU
RgB
RgC
RgD
AD1
AD1
PWM
OUT
8
4
2
1

8.15 - Programming Pages

8.16 - Links

A link to Burkhard Kainka's website, where it all started:
http://www.elektronik-labor.de/Projekte/TPS5.html

If you want more pre-programmed MyCo chips, search for TPS in the search box, top left at http://www.ak-modul-bus.de/cgi-bin/iboshop.cgi?search,0

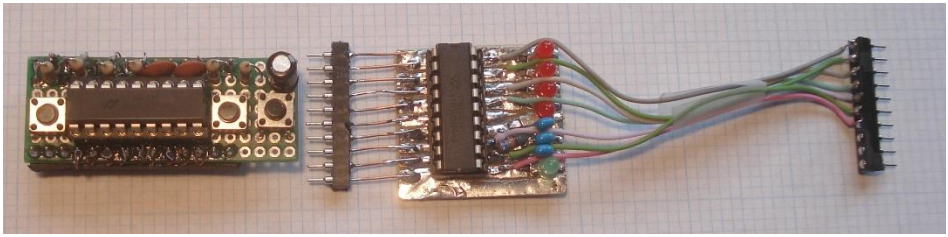
Published eBook version: MyCo_ebook_v16_2014_05_08

This is probably one of the few computers that you can program "handsfree". We tried quickly a version where we used 3 foot switches in parallel to the existing S1, S2 and Reset – and it worked.
Needs getting used to though.

Just in time some other ways to build MyCo:

Version 1: A smaller version, wired underneath with copper wire

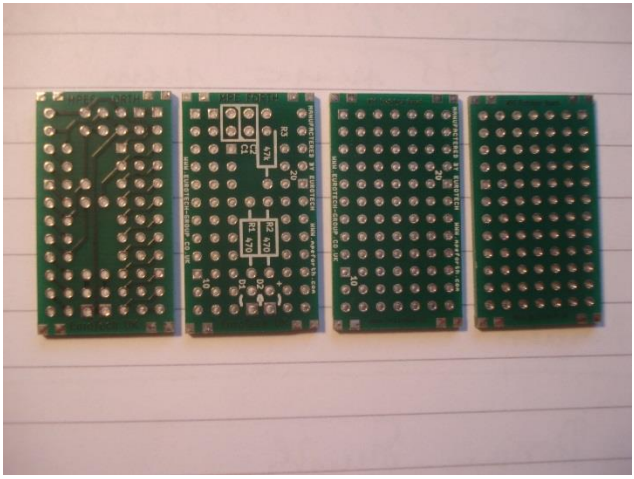
Version 2: A "Dremel" Version, no holes, ready to be integrated into a project. Using an engraving tool to make the PCB on the top side. Thanks Ralf for the idea and the video to prove it works.



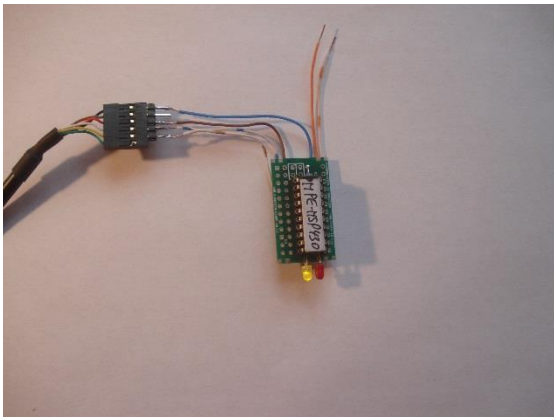
There will be versions with code for other microcontrollers to add to this Holtek version; we are aiming for Atmel, Microchip and TI MSP430 processors for now. Have a look at www.exemark.com over the next couple of months and look for additional information.

The C version is done

A Forth Version is in production and the whole code will be published when stable enough.
Running in a 20 pin MSP430G2553



Here the two boards:
On the left PCB MSP430
On the right ProtoMini



And here the board populated and running via an FTDI USB-to-Serial cable. Forth will be on the chip, MyCo application will be on the chip.

Thanks for reading our eBook, and we hope you had some fun and might actually be tempted to spend the little money to buy a kit and see it working. We might do an additional eBook later if enough new material is available.
So we might meet again.

Short version done 2014_10_18

####