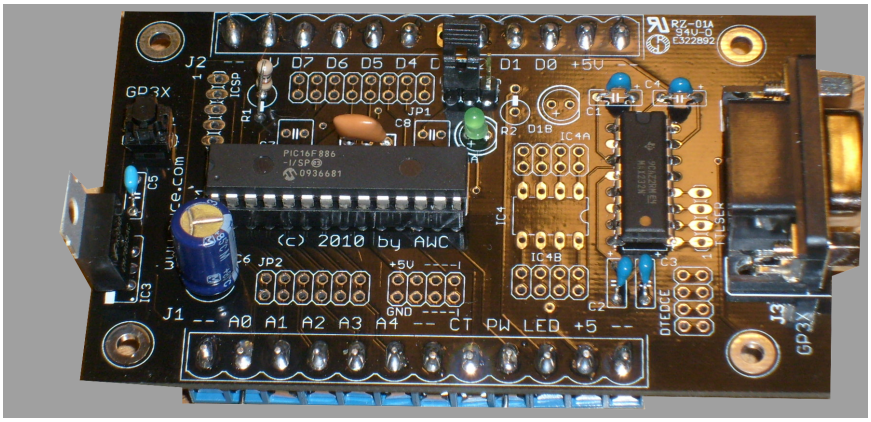
AWC

GP3 Assembler



For the latest documentation see http://www.awce.com/doclib.htm#gp3

AWC
http://www.awce.com
v1.0 11 June 2010

Table of Contents

Overview

The GP-3 can be used in two distinct modes. First, you can use the GP-3 with a PC. The PC communicates with the board via USB or serial port and the board serves as a multipurpose digital and analog I/O board. The second common technique is to use GP3EZ. This software lets you build your program using a series of point and click screens. You can test your program over the PC to GP-3 connection. When you are satisfied, GP3EZ can download the program to the GP-3. When you reset the board with the PGM/RUN jumper set to RUN the board will execute the stored codes even with no PC attached.

This makes the GP-3 very flexible. However, you may not always want to use GP3EZ for every task. First, GP3EZ is easy to use, but it does not completely expose all the features of the GP-3. Second, you may simply want more control over the flow of execution of your program. In these cases, you can use GP3ASM – an assembler for the program codes uses by the GP-3.

Keep in mind that the GP-3 is not a general-purpose CPU. The command set is limited and geared at supporting GP3EZ operations. If you are connected to a PC anyway, using the GP-3 in the tethered mode with a PC-based language (or GP3EZ) is almost certainly a better idea. However, consider a common scenario where GP3ASM can help you.

The GP-3 has several A/D converters. How fast can you read one channel? That depends. Of course, the command to read the A/D convert takes a certain amount of time to execute. In addition, you must send the command to the board at 57600 baud. So even a single byte command takes a certain amount of time to transmit. Then the answer (two bytes) must be transmitted as well.

There is a repeat command so you can ask the board to read many samples with one (slightly longer) command. But that requires you to read the samples and then start a new cycle. You can't just have samples come in forever at a fixed rate using the two conventional techniques. GP3EZ has a method for sending data via the serial port in ASCII, but that's not very efficient compared to the normal way of doing things.

However, with GP3ASM you can write a simple program to sample the A/D port, write the data out in the normal GP-3 format, and repeat. You'll have to be careful, of course, to synchronize the data (for example, ensure the board is not turned on until after the PC software is running and connected). With this scheme, analog data flows in an efficient way, at a steady rate, until the board is reset or turned off.

This document will show you how to use GP3ASM for these types of tasks. It will also discuss how to program your GP-3 hardware with the resulting program.

Documentation

To ensure you have the latest documentation, please check online (http://www.awce.com/doclib.htm#gp3) for the latest versions and updates.

If You Need Help

If you require assistance, please feel free to contact us. The best way to get support is via our FAQ system (http://www.awce.com/faqs). Be sure to check out our Web page for updates at www.al-williams.com/awce.

Details

The GP3ASM software is a hosted service available for free from AWC. If you want to host your own version, it is possible, but requires a Linux or Linux-like environment (e.g., Cygwin) and several other tools (gawk, gcc, etc.). This document assumes you are using the hosted version of the software.

Every program starts with a BEGIN directive and finishes with an END directive. Like a standard assembler, lines can have labels that end with colons. You can use C syntax for any compile-time expressions and C preprocessor symbols with an extra # sign. Comments start with semicolons For example:

```
; Example program
##define MYOUTPUT 6
; Quantum time for delay in milliseconds
##define QUANTUM 100



      BEGIN
top:  HIGH MYOUTPUT
      DLY QUANTUM*2    ; pause a bit
      LOW MYOUTPUT
      JMP top
      END
```

This simple program turns output 6 on, delays for a bit, turns the same output off and jumps back to the top of the program (causing a very high duty cycle square wave to appear on pin 6.

Given a program, visit http://www.awce.com/gp3asm and paste your code into the text box there. Also give your program a name (a simple identifier with no special characters or spaces will work best). Press submit. If your browser downloads a zip file, the compile was successful. If the compile was not successful, you will see an error message. Do not enter text directly into the box since some browsers will lose the text after an error.

If successful, the zip file will contain 3 files. A “read me” file, your source code (with a .gpa prefix), and a binary file with the GP-3 codes in it (a .bin file). Use any zip file extraction utility to extract the .bin file (the binary file).

With the resulting binary file, you simply need to upload it using any standard terminal program to the GP-3 board. Be sure the PGM/RUN jumper is in the PGM position and the terminal program is set to 57600 baud, 8 bits, 1 stop bit, and no parity. You should have hardware handshaking set.

Use your terminal program's send file capability to upload the binary file to the board. Do not use a special protocol (like XModem). Even though the file is binary, you may need to use your terminal's “send text file” or similar function.

Reference

Notes:

- *Standard C language syntax is usable for the value (e.g., 'X', 0x20, 0377, or 0x40-3 are all legitimate values.*

- *Many commands require ARG to be set. However, you may be able to "reuse" the existing value of ARG to save code size and execution speed. Therefore, select commands have two forms. For example, FREQ automatically sets ARG to be the duration of the generated frequency. FREQA is the same command but assumes ARG is already set.*
- *Command tokens (like BEGIN or CHECK) are not case sensitive. Labels and other symbols are.*
- *Skip values can only be positive and only range from 0 to 31. GP3ASM accepts a label for the skip argument and computes the correct value. However, if you try to jump backwards or too far forward, GP3ASM will issue an error message.*
- *Most of these commands are available when the GP-3 is tethered to a PC. If you are not familiar with the GP-3's general command set, please refer to the GP-3 documentation.*

A2D channel

Read the specified A/D channel (0-4). Note: this command causes serial data to be sent just like the standard GP-3 command. Therefore, you probably won't use this command unless you are writing custom software on the remote computer to decode the output.

ACONF justify, reference

Configure the analog channels. If justify is zero, left justify analog values, otherwise right justify (default). The reference can be 0 (default) for all channels are regular analog channels, or use 1 to use channel 3 as the positive reference. If reference is set to 8, then channel 3 is the positive reference and channel 2 is the negative reference. See the GP-3 manual for more information on using the channels as references.

ARGB value

This sets the ARG register in the GP-3 to a byte-sized value (0-255). See **DB** for notes on the format of the value (essentially, a C-language constant). You should rarely need to use this as the GP3ASM opcodes set ARG when necessary automatically.

ARGINC

Increment the ARG register (see ARGB for more information about the ARG register).

ARGDEC

Decrement the ARG register (see ARGB for more information about the ARG register).

ARGW value

This sets the ARG register in the GP-3 to a word sized value (0-65535). See **ARGB**, above, for more details.

BEGIN

Starts every GP3ASM program. You may only use this at the start of your program. No GP3ASM statements (that is, only comments and preprocessor statements) can appear before the **BEGIN**.

CALL label

Calls a labeled subroutine (also see RTN). The return stack is 8 levels deep. Unlike a skip, the CALL can go to any address in your program.

CHECK address, mask, match, skip

Read a value from the 16 word (a word is two bytes) GP-3 RAM given by address, and it with the mask and check it against match. If there is a match, continue. Otherwise skip to the label indicated to find the next instruction to execute. Note that address zero is the previous step value (that is, the last value input; see the GP3EZ help for more details). GP3EZ uses address 1 to manage states, if enabled, but that's only

a convention; the GP-3 does not manage states internally, and any state management is up to your program.

COUNT pin, duration

Count transitions on the indicated pin for the indicated number of milliseconds. The GP-3 samples the pin every 4uS. Note: this command causes serial data to be sent just like the standard GP-3 command. Therefore, you probably won't use this command unless you are writing custom software on the remote computer to decode the output.

COUNTA pin

The COUNT command that uses the ARG register for the duration (see COUNT). Note: this command causes serial data to be sent just like the standard GP-3 command. Therefore, you probably won't use this command unless you are writing custom software on the remote computer to decode the output.

COUNTER

Read the hardware counter and reset it to zero (contrast to COUNTERX). Note: this command causes serial data to be sent just like the standard GP-3 command. Therefore, you probably won't use this command unless you are writing custom software on the remote computer to decode the output.

COUNTERX

Read the hardware counter and do not reset it to zero (contrast to COUNTER). Note: this command causes serial data to be sent just like the standard GP-3 command. Therefore, you probably won't use this command unless you are writing custom software on the remote computer to decode the output.

DB value

Define a “byte” (note that GP3ASM's unit of storage is 14 bits, so the value argument can range to 16383.

DLY value

Delay execution for a specified number of milliseconds.

EER address

Read EEPROM at the specified address (0-FF). Note: this command causes serial data to be sent just like the standard GP-3 command. Therefore, you probably won't use this command unless you are writing custom software on the remote computer to decode the output.

EERA

Read EEPROM from the address specified in ARG. Note: this command causes serial data to be sent just like the standard GP-3 command. Therefore, you probably won't use this command unless you are writing custom software on the remote computer to decode the output.

EEW address, byte

Write a byte to the EEPROM at the location specified by address. Note that addresses greater than FF00 point to the GP-3's 16 word (32 byte) RAM.

EEWA byte

Write a byte to the EEPROM at the location specified by the ARG register.

END

Ends every GP3ASM program. Nothing should be after this line (comments would be acceptable, however).

FREQ pin, frequency, duration

Output simulated sine wave on the indicated digital pin at the specified frequency (in Hertz) for the given duration (milliseconds).

FREQA pin, frequency

Output simulated sine wave on the indicated digital pin at the specified frequency (in Hertz) for the duration in milliseconds specified by ARG.

GETDIR

Get the data direction register (the bit pattern that indicates if digital pins are inputs or outputs). Note: this command causes serial data to be sent just like the standard GP-3 command. Therefore, you probably won't use this command unless you are writing custom software on the remote computer to

decode the output.

GETIN

Get the digital input bits as a byte. Note: this command causes serial data to be sent just like the standard GP-3 command. Therefore, you probably won't use this command unless you are writing custom software on the remote computer to decode the output.

HIGH pin

Set the indicated pin to a high logic level.

HRESET

Hard reset to reset conditions.

INANALOGC channel, cc, channel2, skip

Read analog channel2 once. Then compare it to the channel using the condition code cc (see below). If the comparison fails, skip to the indicated label. Otherwise, simply continue.

INANALOGCW channel, cc, channel2

Read analog channel2 once. Then compare it to the channel using the condition code cc (see below). Wait until the comparison succeeds before continuing execution.

INANALOGX channel, cc, value, skip

Read the analog channel and compare it to the value using condition code cc (see below). If the comparison fails, skip to the indicated label to find the next instruction. Otherwise, simply continue.

INANALOGXW channel, cc, value

Read the analog channel and compare it to the value using condition code cc (see below). Wait for the comparison to succeed.

INCT match, skip

Test the hardware counter against the match value. If the counter is greater than the match value, proceed. Otherwise skip to the indicated label for the next instruction. The counter is not cleared. Note that if the counter overflows, the value may appear less than the count. That is overflow is not detected.

INCTW match

Test the hardware counter against the match value. Do not proceed until the counter is greater than the value. See INCT for details. The counter is not cleared.

INP pin

Read the indicated pin and output an ASCII 0 (30 hex) or 1 (31 hex) on the serial port. Note: this command causes serial data to be sent just like the standard GP-3 command. Therefore, you probably won't use this command unless you are writing custom software on the remote computer to decode the output.

INPUT mask, match, skip

Read the digital inputs, and them with the mask value, and compare them to the match value. If they do not match, skip to the provided label to find the next instruction. Otherwise, continue.

INPUTW mask, match

Read the digital inputs, and them with the mask value, and compare them to the match value. Wait until a match occurs.

JMP label

Jump to a new instruction provided by the indicated label. Unlike a skip, the JMP can go to any address in your program.

LED state

Set the state of the onboard LED (0 or 1). You may wish to use the predefined ON/OFF constants with this command.

LOW pin

Set the indicated pin to a logic low level.

NEXT loop, label

Continue the indicated loop (i, j, or k; see below) by decrementing the loop count. If the count is zero, the next instruction executes. Otherwise, execution continues at the indicated label. Note this label is not

limited as a skip value (that is, it can jump to any location in your program).

NOP

No operation.

PULSEIN pin, state

Read a pulse time in 2uS units (see the GP-3 manual for details). Note: this command causes serial data to be sent just like the standard GP-3 command. Therefore, you probably won't use this command unless you are writing custom software on the remote computer to decode the output.

PULSEOUT pin, duration

Invert the state of the indicated pin, for the specified duration (in 2uS units).

PULSEOUTA pin

Invert the state of the indicated pin, for the duration specified in the ARG register (2uS units).

PWM pin, duty_cycle, duration

Generate PWM on the specified output, with the given duty cycle and duration (in milliseconds).

PWMA pin, duty_cycle

Generate PWM on the specified output with the given duty cycle. The duration in milliseconds is taken from the ARG register.

RCTIME pin, state

See the GP-3 manual for more on this command. Note: this command causes serial data to be sent just like the standard GP-3 command. Therefore, you probably won't use this command unless you are writing custom software on the remote computer to decode the output.

REPEAT count

Set the repeat count which affects certain commands. See the GP-3 manual for more information on repeating.

RST

Reset the GP-3's communication state (contrast with HRESET).

RTN

Return from a subroutine.

SEND byte

Send a byte over the serial port. Then send the current step value as a 4 digit hex number in ASCII followed by a carriage return. For example, if byte is “X” and the step value is 03FF hex, the GP-3 would output 6 bytes: An X, a zero, a three, two “F” characters, and a carriage return.

SERANY skip

Check for any serial character (the GP-3 stores the last character sent to it). If there is a character waiting, consume it and continue. Otherwise, go to the skip label.

SERANYW

Wait for any character before proceeding. The character is consumed.

SERIN match, skip

Look for a particular match character from the serial port. If there is no character, or the character does not match, go to the skip label. Note that if the character does not match, it is not consumed and will be available for future SERIN and related commands. If the character matches, it is consumed and execution continues.

SERINW match

Wait for the specified match character.

SETCTR prescale, source

Set the hardware counter options. The prescale may be 0 for 1:1, 1 for 1:2, 2 for 1:4, or 3 for 1:8. The source may be zero (internal 5MHz clock) or one (the CT pin on the GP-3 board). This instruction also clears the counter.

SETDIR byte

Set the data direction byte. Each bit controls the input/output state for the digital pins. For example, a 1 in bit 0 makes digital pin 0 an input. A zero makes the pin an output. By default all pins are inputs until

you either change them or use a command that sets them to outputs (e.g., HIGH or LOW). See the GP-3 manual for more details.

SETLOOP loop, initial

Initialize the indicated loop (i, j, or k; see below). Combined with NEXT this allows you to create up to three countdown loops.

SETOUT byte

Sets the output bits on all digital pins at once. Note this command does not force any pins to output state (see SETDIR).

TOG pin

Toggle the indicated pin so that if it was high, it will be set low or vice versa.

VERSION

Return the GP-3 version (currently 33 hex) over the serial port. Note: this command causes serial data to be sent just like the standard GP-3 command. Therefore, you probably won't use this command unless you are writing custom software on the remote computer to decode the output.

VERSION2

Secondary version. Hex 34 indicates a limited production version that has a smaller program space. Most GP-3s should respond with a hex 35 for this command. Note: this command causes serial data to be sent just like the standard GP-3 command. Therefore, you probably won't use this command unless you are writing custom software on the remote computer to decode the output.

XPWM frequency, duty_cycle

Set the hardware PWM output to the specified frequency and duty cycle (see the GP-3 manual for more details).

XPWMA duty_cycle

Set the hardware PWM output the specified duty cycle using the frequency in ARG.

Predefined Constants

ON

A one value. Used with LED or anywhere you want a descriptive 1.

OFF

A zero value. Used with LED or anywhere you want a descriptive 0.

i

The first loop counter (see SETLOOP and NEXT).

j

The second loop counter (see SETLOOP and NEXT).

k

The third loop counter (see SETLOOP and NEXT).

lt

Condition code for analog commands. Less than.

eq

Condition code for analog commands. Equal to.

gt

Condition code for analog commands. Greater than.

le

Condition code for analog commands. Less than equal to.

ge

Condition code for analog commands. Greater than equal to.

About State

If you use states in GP3EZ (which are disabled by default) you may wonder why there are no commands related to state management. That's because GP3EZ creates state management in the compiled code using the commands shown. Here's how it works.

The GP-3 has an internal 16 word (32 byte) RAM. The chip uses the first word to store the “step value” for most commands. The other words are available for any use. GP3EZ uses the first word to store the present state. You can access the RAM using EEW and EER with addresses starting at FF00 hex. Note that's byte access, so to get the entire state value, for example, you'll need to read FF00 and FF01. FF00 has the high byte and FF01 has the low byte.

When using states, the compiled GP3EZ software includes a CHECK instruction before each step that tests for the allowed states (stored as a bit pattern). If the CHECK fails, it skips over to the next command. Then at the end of a statement, the GP3EZ compiler inserts code to change the current state if necessary (an EEW command).

Naturally, you can implement states in many other ways and this is just how GP3EZ does it. The actual GP-3 chip has no knowledge of states, so you are free to implement them as you fit if you want to use them.

Using Macros

You can use C language constructs by prefixing them with the # sign. For example, to create 10 ARGINC instructions:

```
# for (int i=0; i<10; i++) {
ARGINC
#  }
```

Examples

Hello World (blinking LED)

For a first example, consider the code below. It flashes the onboard LED 10 times, pauses, and then repeats. It shows how to use the LED, delays, and looping.

```
      BEGIN       ; all programs use a BEGIN
top:  SETLOOP i,10      ; set the “i” loop to run 10 times
loop1:
      LED ON            ; turn on the LED
      DLY 500           ; pause for 500ms
      LED OFF           ; LED off
      DLY 500           ; another 500ms delay
      NEXT i, loop1     ; execute the loop
      DLY 5000          ; after loop, pause for 5 seconds
      JMP top           ; do it all over again
      END
```

Dealing with positive skips

Skips can only be small and positive, but sometimes you need to go backwards or far away. You have to write multiple instructions. For example, Suppose you want to wait for a the user to press any key on the attached serial terminal, but you also want to beep until the key is pressed. The logic would be: beep, check for a character, if no character, go back to the start. But because the skip for SERANY can only go forward:

```
##define SPEAKER 0
      BEGIN
; beep and wait for keypress
beepwait:
      FREQ  SPEAKER, 800, 500    ; note, preloading ARG and using FREQA would be smart
her
      SERANY gotkey
      JMP beepwait
gotkey: FREQ SPEAKER,1200,500  ; confirmation beep
halt: JMP halt
      END
```

Custom Data Logging

One question that often comes up is: “How fast can I read the GP-3's A/D converter?” The answer depends on lots of factors, but in general, using the GP-3 as it is intended means you have to send a serial byte to the board and then receive 2 more back for each conversion. So regardless of how fast the converter itself is, you can't get any faster than the time it takes for 3 characters to go across the PC interface. If you only need a certain number of samples, you can use the REPEAT command to cut that overhead significantly. You send a few bytes to set up the transfer and after that it takes 2 byte-sized sends plus the converter overhead to get the specified number of samples.

But what if you want to repeat forever? Or maybe you want to read two channels. GP3ASM can create a custom program that does what you want. The idea is to just make the GP-3 send data continuously. One issue, however, is the PC has no easy way to synchronize on the data stream. It is difficult to tell which byte is the first part of the first reading and which byte is the second part of another reading.

However, presuming the PC can keep up with the data stream (which is slow compared to a modern PC), one scheme would be to have the PC send a single command to get the GP3 started. Here's a simple program:

```
      BEGIN
      SERANYW   ; wait for any input on the serial port
adloop:
      A2D 0
      JMP adloop
      END
```

This program will wait for any character to arrive on the serial port and then send data from A/D channel 0 as

fast as possible. If you wanted, for example, the digital data also you could write:

```
      BEGIN
      SERANYW   ; wait for any input on the serial port
adloop:
      A2D 0
      GETIN    ; read digital inputs all at once
      JMP adloop
      END
```

Now your program would have to receive 3 bytes. Of course, you could also use DLY commands if you wanted to work at a particular rate. In addition, although it would slow down the loop, you could test for particular input characters to start and stop the flow of data.