Sun, 2020-06-28 11:21 — Costin G

Hi,

Sorry for missing this, I didn't spot your question. Yes, I'm still using the shield and it works just fine. With this modification of the code DCC++ BaseStation code:

<https://github.com/costing/BaseStation>

At power on it tries a few times in quick iterations to bring the power on, to allow the keepalive capacitors to charge. Any short afterwards keeps the power off until explicitly turned back on.

For all these to work you need to connect the SF pin of MC33926 to pin 4 of Arduino, in addition to the other pins as per the documentation.

I don't have any issue reading or programming CVs either.

The only open question is the frog juicers. I'm running a 3-rail system so I don't have any such hardware around. If you manage to find a way around this I'd be curious to hear about it.

Cheers,

.costin

DCCpp\_Uno/ErrorMonitor.cpp

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  |  |
|  | ErrorMonitor.cpp |
|  | COPYRIGHT (c) Costin Grigoras |
|  |  |
|  | Part of DCC++ BASE STATION for the Arduino / extended for Pololu shield |
|  |  |
|  | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
|  |  |
|  | #include "DCCpp\_Uno.h" |
|  | #include "ErrorMonitor.h" |
|  | #include "Comm.h" |
|  |  |
|  | /////////////////////////////////////////////////////////////////////////////// |
|  |  |
|  | long int lastPowerOffTime = 0; |
|  |  |
|  | //#define ERROR\_MONITOR\_DEBUG true |
|  | //#define ERROR\_MONITOR\_DEBUG\_FINE true |
|  |  |
|  | #define MILLIS\_TO\_RECOVER\_MAX 10 |
|  | #define ACT\_UP\_TO 100 |
|  |  |
|  | void errorCheck() { |
|  | if (digitalRead(SIGNAL\_ENABLE\_PIN\_PROG) == LOW) |
|  | lastPowerOffTime = millis(); |
|  |  |
|  | if (digitalRead(ERROR\_MONITOR\_PIN) == 0 && digitalRead(SIGNAL\_ENABLE\_PIN\_PROG) == HIGH) { |
|  | digitalWrite(SIGNAL\_ENABLE\_PIN\_PROG, LOW); // disable both Motor Shield Channels |
|  | digitalWrite(SIGNAL\_ENABLE\_PIN\_MAIN, LOW); // regardless of which caused current overload |
|  |  |
|  | if (millis() - lastPowerOffTime < ACT\_UP\_TO) { |
|  | // at startup time large capacitors charging simultaneously can trigger the error state |
|  | // re-enable the power for a few milliseconds to allow them to charge |
|  |  |
|  | const long int recoveryStarted = millis(); |
|  |  |
|  | long lastRecoveryAttempt = 0; |
|  |  |
|  | while (millis() - recoveryStarted < MILLIS\_TO\_RECOVER\_MAX) { |
|  | lastRecoveryAttempt = millis(); |
|  |  |
|  | digitalWrite(SIGNAL\_ENABLE\_PIN\_MAIN, LOW); |
|  | digitalWrite(SIGNAL\_ENABLE\_PIN\_PROG, LOW); |
|  |  |
|  | digitalWrite(SIGNAL\_ENABLE\_PIN\_MAIN, HIGH); |
|  | digitalWrite(SIGNAL\_ENABLE\_PIN\_PROG, HIGH); |
|  |  |
|  | while (digitalRead(ERROR\_MONITOR\_PIN) == 1 && (millis() - recoveryStarted < MILLIS\_TO\_RECOVER\_MAX)) ; |
|  |  |
|  | if (digitalRead(ERROR\_MONITOR\_PIN) == 0) { |
|  | #ifdef ERROR\_MONITOR\_DEBUG\_FINE |
|  | INTERFACE.print("\nRetry @ "); |
|  | INTERFACE.println(millis() - recoveryStarted); |
|  | #endif |
|  | } |
|  | else { |
|  | #ifdef ERROR\_MONITOR\_DEBUG |
|  | INTERFACE.print("\nWorked@"); |
|  | INTERFACE.println(millis() - lastRecoveryAttempt); |
|  | #endif |
|  |  |
|  | break; |
|  | } |
|  | } |
|  |  |
|  | #ifdef ERROR\_MONITOR\_DEBUG |
|  | if (digitalRead(ERROR\_MONITOR\_PIN) == 0) |
|  | INTERFACE.println("\nRecovery didn't work"); |
|  | #endif |
|  |  |
|  | // let it process messages and maybe try again later, or show the error if not |
|  | return; |
|  | } |
|  | #ifdef ERROR\_MONITOR\_DEBUG |
|  | else |
|  | INTERFACE.println("\nNo recovery"); |
|  | #endif |
|  |  |
|  | if (digitalRead(ERROR\_MONITOR\_PIN) == 0) { |
|  | #ifdef ERROR\_MONITOR\_DEBUG |
|  | INTERFACE.print("\nShort circuit\n"); |
|  | #endif |
|  |  |
|  | INTERFACE.print("<p2>"); // print corresponding error message |
|  | } |
|  | } |
|  | } |